AD-A051 886 DAYTON UNIV OHIO RESEARCH INST F/G 19/5
FIRE CONTROL SYSTEM ANALYSIS VOLUME II. COMPUTER PROGRAMMING TA--ETC(U)
NOV 77 C KING F33615-77-C-1056
UNCLASSIFIED AFAL-TR-78-16-VOL-2 NL

AFAL-TR-78-16
Volume II

AD-A051886

FIRE CONTROL SYSTEM ANALYSIS

Volume II - Computer Programming Tasks

Prepared By:

University of Dayton
Research Institute
Dayton, Ohio 45469

November 1977

Technical Report AFAL-TR-78-16, Vol II

Final Report for Period 1 Nov 76 - 30 Sep 77

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Dennis Adamiec, Capt Arthur A. Duke Jr

FOR THE COMMANDER

Marion Spector

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/RWT, *W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAL-TR-78-16, Volume II
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: Fire Control System Analysis
Volume II - Computer Programming Tasks

5. TYPE OF REPORT & PERIOD COVERED: Final Report 11/1/76-9/30/77

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Carl King

8. CONTRACT OR GRANT NUMBER(s): F33615-77-C-1056

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
University of Dayton Research Institute
Dayton, Ohio 45469

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63605F/76290359

11. CONTROLLING OFFICE NAME AND ADDRESS:
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

12. REPORT DATE: November 1977

13. NUMBER OF PAGES: 164

14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*:

15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*:
This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:

18. SUPPLEMENTARY NOTES:

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:
Kalman Filter, Target State Measurement, Radar Lag, Multiple Digital Scan Converter, Air Combat Evaluation Algorithm

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*:
The following three tasks are included in Volume I. Gaussian noise added to true range generated by various kinds of simulated trajectories served as "measurements" which were input to several kinds of Kalman filters. Filter output is compared graphically and by rms deviations. Independent methods were devised for evaluating the accuracy of target state vectors (position, velocity, and acceleration) obtained from a director fire control system.
(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (Continued)

Attempts to find an adequate software correction to the radar lag were not successful. Analysis of data on Sight Eval tapes casts doubt on its validity.
Volume II contains reports on the following tasks. The equations used in the LCOS, TRACER, and ACE algorithms were rewritten such that they could be performed on the MDSC computer. The ACE algorighm was modified for implementation on the ROLM 16/64 computer.

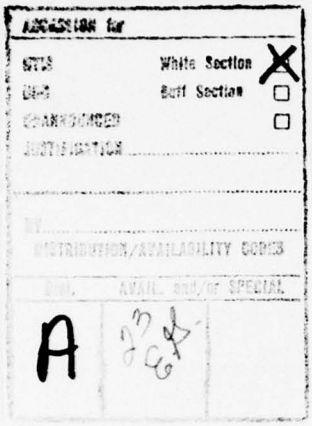


TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF ILLUSTRATIONS

SECTION 1
BACKGROUND

The recognition by the Air Force Avionics Laboratory (AFAL) of the need for improved sensors and techniques for implementing fire control director algorithms has resulted largely from the conclusions of two programs - EXPO V and Fire/Fly. EXPO V is the latest in a series of man-in-the-loop simulation studies of new fire control and gunsight concepts. The Fire/Fly program is the integrated Fire Control/Flight Control study which is investigating methods for integrating the fire control system with the flight control system to improve effectiveness while increasing survivability. Both programs have cited the necessity for improved sensors as well as improved director algorithms.

The purpose of the Fire Control Systems analysis effort, summarized in the following report, was to examine existing fire control systems test data (Sight Eval), identify error sources, evaluate and modify fire control algorithms, and perform programming for simulation, weapon system investigation, and validation.

The tasks contained in this volume were for computer programming support.

SECTION 2
TASK DEFINITION

The broad categories of effort contained in the Statement of Work were refined by coordination with Captain J. Silverthorn, RWT-2, and resulted in the following series of tasks.

2.1 REHOST MODULAR DIGITAL SCAN CONVERTER (MDSC) CROSS ASSEMBLER

The Hughes Aircraft Company Modular Digital Scan Converter (MDSC) forms an important part of the director gunsight flight test. It will be performing important computations and Heads Up Display (HUD) symbol generation. It is imperative that a capability exist at Tyndall Air Force Base (TAFB) to modify the MDSC assembly language program in order to change computations or symbols. To accomplish this, an existing MDSC cross assembler will be rehosted on either a PDP-11 or ROLM 16/64 computer, or both.

2.2 DEVELOP AN AERIAL COMBAT EVALUATOR (ACE) ALGORITHM

An Aerial Combat Evaluator (ACE) algorithm was used on the Sight Eval program and shown to be valuable. It indicates to the pilot the expected number of bullets that would have hit the target had he fired the gun. As a result, it provides feedback even during "dry run" passes. The algorithm used in Sight Eval and the algorithm developed in a previous contract will be compared, the best selected, and then implemented on the ROLM computer.

2.3 FIRE CONTROL DOCUMENTATION

A significant amount of documentation of previously developed algorithms needs to be performed. Many of these algorithms are being used in the director flight test.

SECTION 3

PROGRAM SUPPORT

3.1 REHOST MODULAR DIGITAL SCAN CONVERTER (MDSC) CROSS ASSEMBLER

The Modular Digital Scan Converter (MDSC) computer was developed by Hughes Aircraft Company for use in airborne fire control systems. Since the MDSC computer does not have an assembly language compiler, programming is done through a cross assembler. (A cross assembler is a program that generates machine code for a given computer from the assembly language of another computer.) The cross assembler for the MDSC computer was initially hosted on the Sigma V computer at Hughes and all programming had to be done through this system. To expedite work on programming on the MDSC computer, the AFAL of Wright-Patterson Air Force Base (WPAFB) contracted with the University of Dayton Research Institute (UDRI) to rehost the MDSC Cross Assembler to the PDP-11 computer and the ROLM 16/64 computer. These computers are on-site at TAFB where the Cross Assembler program is to be implemented.

3.1.1 Approach

The rehosting process is illustrated in Figure 1. The first step in rehosting the FORTRAN coded MDSC Cross Assembler to the PDP-11 and ROLM 16/64 computers was to rehost it to the CDC computer at WPAFB. The step of rehosting the MDSC Cross Assembler to the CDC computer system was taken because of the familiarity and availability of this system to the UDRI personnel. An added advantage of this step was that the CDC system checked the operation of all subsections of a program. Once the MDSC Cross Assembler was operational on the CDC system, work was directed to the task of rehosting it on the PDP-11 and ROLM 16/64 computers.

The PDP-11/55 computer available at TAFB has the RSX-11M operating system and a FORTRAN IV-PLUS compiler. UDRI

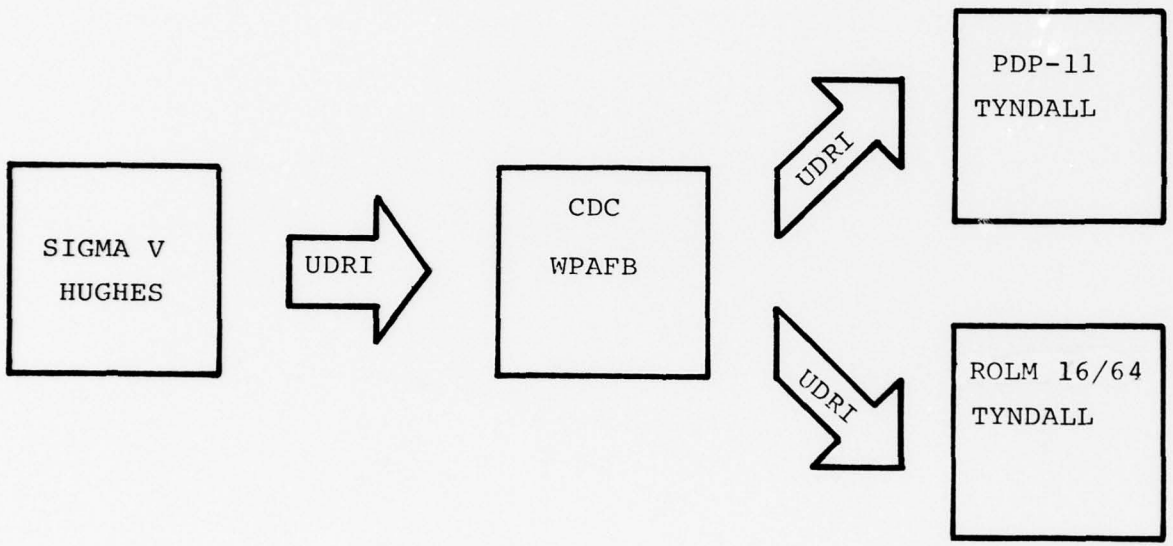


Figure 1. Approach to Rehosting

was unable to locate a PDP-11/55 computer at WPAFB, but a PDP-11/45 with the same operating system and compiler was located. This proved to be a satisfactory substitute since the two systems are operationally the same. The rehosting of the Cross Assembler to the PDP-11/45 was completed in August 1977 and sent to TAFB for implementation.

Although the initial plans were to rehost the Cross Assembler to both the PDP-11 and the ROLM 16/64 computers, only the rehosting to the PDP-11 was completed. Rehosting to the ROLM 16/64 computer could not be initiated because of scheduling problems on this computer. The Air Force Flight Dynamics Laboratory (AFFDL) had priority on this computer and its current programs consumed practically all of the available time.

3.1.2 Verification

Verification of the rehosted MDSC Cross Assembler was accomplished by running five documented assembly language programs on each of the versions and comparing the output values to the verification values. Two of the verification programs were written by Hughes (Reference 1) and three by AFAL (Reference 2). Four of the five agreed perfectly with the verification listings. The output of one of the AFAL programs did not agree fully with the expected output. However, this was not considered a serious problem since this program has not been completely debugged. The listing of these five verification programs as compiled by the PDP-11 version of the MDSC Cross Assembler are shown in Appendix A.

3.1.3 Rehost on the CDC

Rehosting the MDSC Cross Assembler program from the Sigma V computer to the CDC computer system required several modifications because of hardware differences. The major difference in the two computer systems is word size. The Sigma V uses a 32-bit word with four bytes per word and the CDC uses a 60-bit word with 10 bytes per word. To circumvent

this problem, Sigma V coding was changed to the CDC format of 10 bytes/word.

Another difference in the two systems is that the Sigma V is a two's complement computer and the CDC is a one's complement computer.[1] It was found that this problem could be overcome by modifying the masking subprogram to make the CDC function as a two's complement computer.

Modifications that had to be made to further make the Cross Assembler compatible to the CDC system were as follows.

(1) A program specification statment had to be added as the first statement of the main program.

(2) Variable and subroutine names had to be reduced to seven or less characters.

(3) The end-of-file check and READ statement had to be modified.

(4) Logical unit numbers for input and output had to be less than 100.

(5) All DO statements had to be changed so that:

a) no computations occurred within the DO statement.
b) all index increments were positive,
c) the index started at an integer greater than or equal to 1.
d) the branch statements back into a DO loop had to be eliminated.

(6) External function names had to be changed to agree with those used in the CDC FORTRAN compiler.

(7) FORMAT statements were changed to make the output more readable.

---

[1] A one's complement computer represents negative integers as the complement bit-by-bit of the positive integers. A two's complement computer represents negative integers as a one's complement with the number one added to the negative integer.

Several definite coding errors were detected in the Cross Assembler program and corrected in the rehosting process. Several of the major errors occurred in the assembly language symbolic names and the machine language for each. These errors were corrected to agree with Reference 1. In addition, the tabbing subprogram code was changed to tab over commas and blanks in the assembly language cards. As a result of rehosting the Cross Assembler program to the CDC, the operation of each subprogram was checked and demonstrated to be accurate.

3.1.4 Rehost on the PDP-11/55

The major changes required to rehost the MDSC Cross Assembler to the PDP-11/55 computer were similar to those required for the rehost to the CDC computer. Machine compatibility was not a problem in this rehosting since the PDP-11/55 computer with the RSX-11M operating system and FORTRAN IV-PLUS compiler mimics the word structure of the Sigma V computer.

In rehosting the CDC compatible Cross Assembler to the PDP-11, the following changes were made.

(1) Variable and subroutine names had to be reduced to six or less characters.

(2) DATA statements had to be altered to eliminate implied DO loops.

(3) The end-of-file check in the READ statement had to be modified.

(4) The FORMAT had to be changed to output a PDP-11 compatible output.

(5) External function names had to be changed to the corresponding names in the PDP compiler.

(6) Multiple assignment statments had to be recoded so that there was no more than one equal sign in each line of code.

(7) A routine had to be coded to handle the decoding of hexidecimal input integers

as characters to the respective numerical integer values used to compute machine language instruction from assembly language input.

(8) DOUBLE PRECISION statements removed in rehosting to the CDC system had to be replaced.

In compiling the Cross Assembler on the PDP-11 computer, it was found that the Cross Assembler program exceeded the capacity of the machine. The maximum capacity of the PDP-11 is 32 K words of 16-bit length. To obviate this problem, the concordance table of assembly language labels was deleted from the program. After the above modifications were made and the program shown to be operational, punched card decks of the Cross Assembler and its verification programs were delivered to AFAL. Appendix B is a listing of the MDSC Cross Assembler program.

3.1.5 References

(1) MDSC Programmer's Reference Manual Volume 4, Hughes Aircraft Company, Culver City, California, HAC Reference no. D2385, March 1976.

(2) AN/UYK-30 Reference Manual, Hughes Aircraft Company, Culver City, California, Report no. P76-148, Ref. A, November 1976.

3.2 AIR COMBAT EVALUATION ALGORITHM

3.2.1 Task Definition

The University of Dayton Research Institute has developed an Air Combat Evaluation (ACE) algorithm to simulate the number of hits scored during air encounters between an attacking aircraft and a target aircraft. This algorithm, when integrated into the flight control system of an attack aircraft, will allow the pilot to experience combat conditions involving evasive maneuvers without using live ammunition. The obvious consequence of the use of this algorithm is improved aircraft-aircraft combat training techniques.

The ACE algorithm uses a statistical approach to compute the expected number of hits scored by the attacking aircraft on the target aircraft. This information is recorded and available to the pilot on a real-time basis. The position of the target aircraft is obtained from the fire control system of the attacking aircraft. Specific inputs to the ACE algorithm are azimuth and elevation of the computed projectile stream at target range with respect to the target.

3.2.2 Algorithm Development

The ACE algorithm is a modification of the Uniform Normal Algorithm (Appendix 1; Ref. 1). ACE is coded in FORTRAN as a subroutine for implementation on the ROLM 16/64 computer. However, because of the unavailability of the ROLM system the major coding effort was done using the CDC computer system.

The geometry for the ACE algorithm is shown in Figure 2. The target plane is normal to the line of sight from the attack aircraft to the target with its origin at the center of the target. The point of intersection of the previous projectile stream (one computer cycle before) with the target plane is represented by the vector $\vec{R}_{mo}$ and the point of intersection of the current

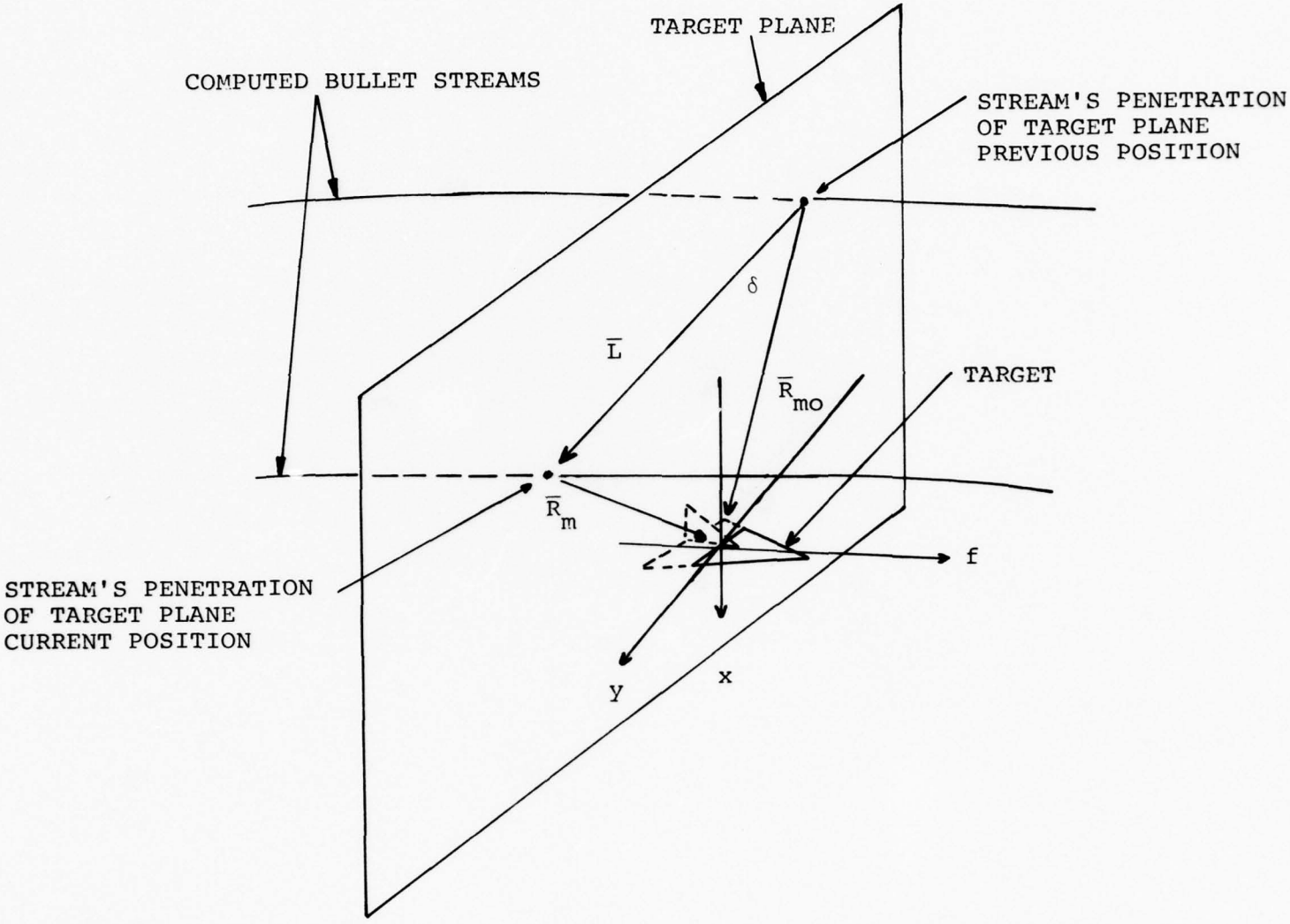


Figure 2. Geometry for ACE Algorithm

projectile stream by $\vec{R}_m$. The relative motion of the projectile stream is represented by $\vec{L}$. The left-handed coordinate system is chosen with the y-axis parallel to $\vec{L}$ and positive x down.

The trajectory of the projectile stream is that of a nominal projectile. The dispersion of projectiles about the projectile stream is assumed to be uniform along $\vec{L}$ and random and normally distributed in the target plane as shown in Figure 2. The density function for the projectile stream is

$$f_{BS}(x,y) = \frac{e^{-\frac{x^2}{2\sigma_B^2}}}{L\sigma_B\sqrt{2\pi}} \quad , \quad |y| \leq L/2 \qquad (1)$$

$$= o \quad , \quad |y| > L/2$$

where σ_B is the standard deviation of this projectile stream, in radians, and L is the magnitude, in radians, of the relative motion vector $\vec{L}$.

The target is represented by a bivariate normal distribution in the target plane as shown in Figure 3. The density function for the target is

$$f(x,y) = e^{-\frac{(x-x_T)^2+(y-y_T)^2}{2\sigma_T^2}} \qquad (2)$$

where σ_T is the standard deviation of the target, in feet, and x, x_T, y, y_T in feet. The standard deviation of the target in radians is then given by

$$\sigma_{TR} = \frac{\sigma_T}{R}$$

Using this notation the density function of the target can be written

$$f_T(x,y) = e^{-\frac{(x-x_T)^2+(y-y_T)^2}{2\sigma_{TR}^2}} \qquad (3)$$

where x, x_T, y, y_T are now also radians.

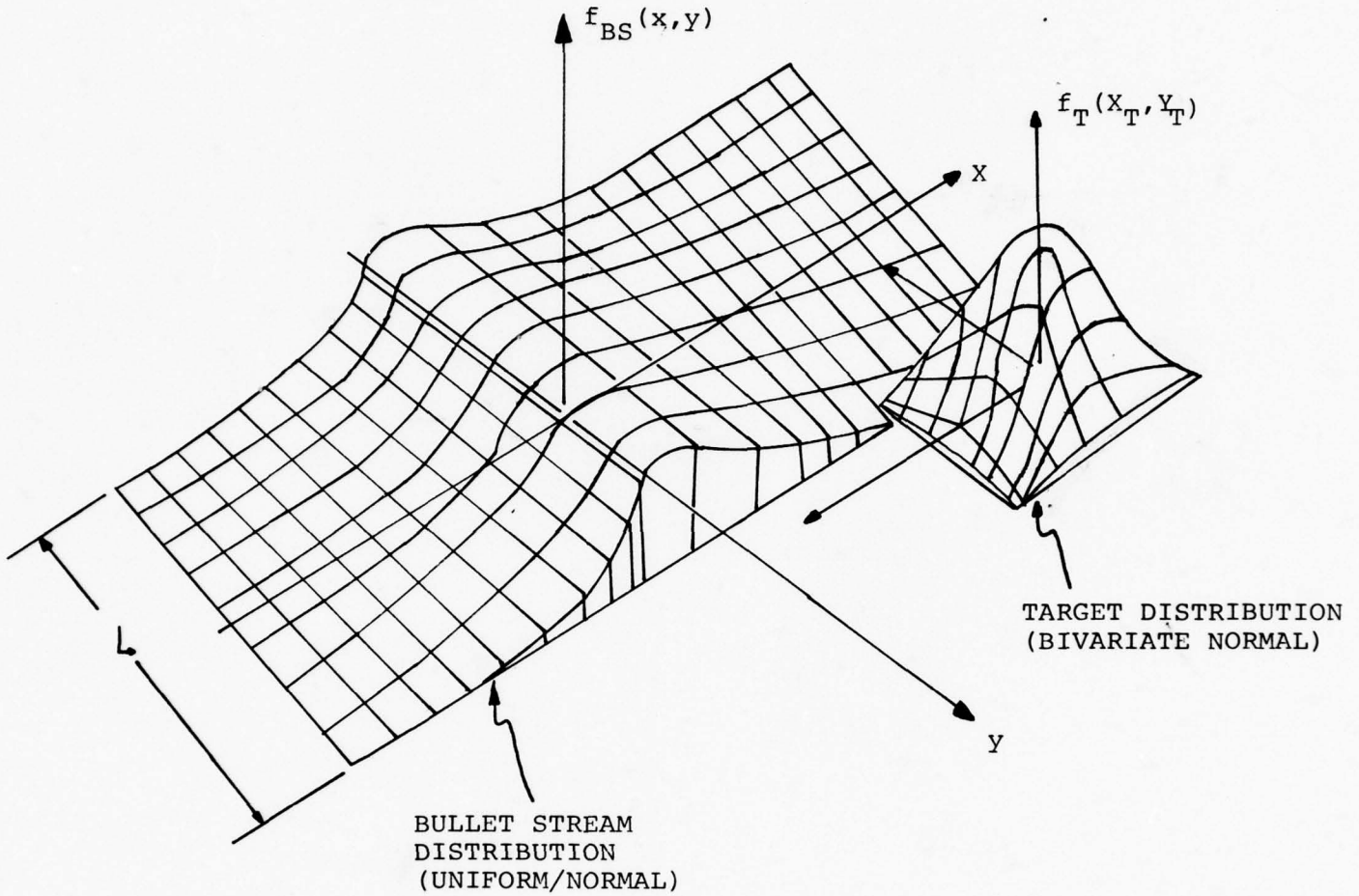


Figure 3. Bullet Stream and Target Distribution Functions

The expected number of hits on the target is given by the product of the number of projectiles in the interval L and the probability that one projectile will hit the target. The probability that one projectile will hit the target is the intersection of the two density functions. Therefore, the expected number of hits is

$$E = N_B \int_{-\infty}^{\infty} \int_{-L/2}^{L/2} f_T(x,y) f_{BS}(x,y) dy\, dx \qquad (4)$$

Preliminary to computing expected number of hits, several important intermediate values must be found. The principle computations are described below.

3.2.3 The Relative Motion Vector

The relative motion vector defines the motion of the projectile stream between the previous position and the current position. The magnitude, L, of this vector is the distance between successive projectile streams at target range measured relative to the target position in the plane. Thus, $\vec{L}$ is given by

$$\vec{L} = \vec{R}_{mo} - \vec{R}_m \qquad (5)$$

where $\vec{R}_{mo}$ is the previous position vector and $\vec{R}_m$ is the current position vector.

3.2.4 The Angle Between the $\vec{L}$ and $\vec{R}_{mo}$

The angle between the vectors $\vec{L}$ and $\vec{R}_{mo}$ is represented by δ in Figure 1 and is computed by the dot product relationship

$$\cos \delta = \frac{\vec{L} \cdot \vec{R}_{mo}}{|\vec{L}|\,|\vec{R}_{mo}|} \qquad (6)$$

3.2.5 The Miss Distance

The miss distance, with components x_T and y_T, is the line in the target plane from the target to the midpoint of $\vec{L}$, as shown in Figure 4. The component x_T is the normal distance from the target to $\vec{L}$ and is given by

$$x_T = |\vec{R}_{mo}|\sqrt{1-\cos^2\delta} \tag{7}$$

The component y_T is along L and is given by

$$y_T = |\vec{R}_{mo}|\cos\delta - |\vec{L}|/2 \tag{8}$$

3.2.6 The Expected Number of Hits

The expected number of hits is calculated from Equation (4) which has been integrated to obtain the following closed form expression

$$E = \frac{N_B \sigma_{TR}^2 \sqrt{2\pi}}{L\sqrt{\sigma_{TR}^2 + \sigma_B^2}} \; e^{-\frac{x_T^2}{2(\sigma_{TR}^2 + \sigma_B^2)}} \left[\operatorname{erf}\left(\frac{\frac{L}{2} - y_T}{\sigma_{TR}}\right) - \operatorname{erf}\left(\frac{-\frac{L}{2} - y_T}{\sigma_{TR}}\right) \right] \tag{9}$$

where N_B is the number of projectiles at target range per computer cycle time (fire rate x cycle time), σ_{TR} is the angular standard deviation of the target (radians), σ_B is the angular standard deviation of the projectile stream (radians), L is the magnitude of the relative motion vector (radians).

3.2.7 The Error Function

The Error Function used in the ACE algorithm is given by

$$\mathrm{ERF}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-u^2/2} \, du \tag{10}$$

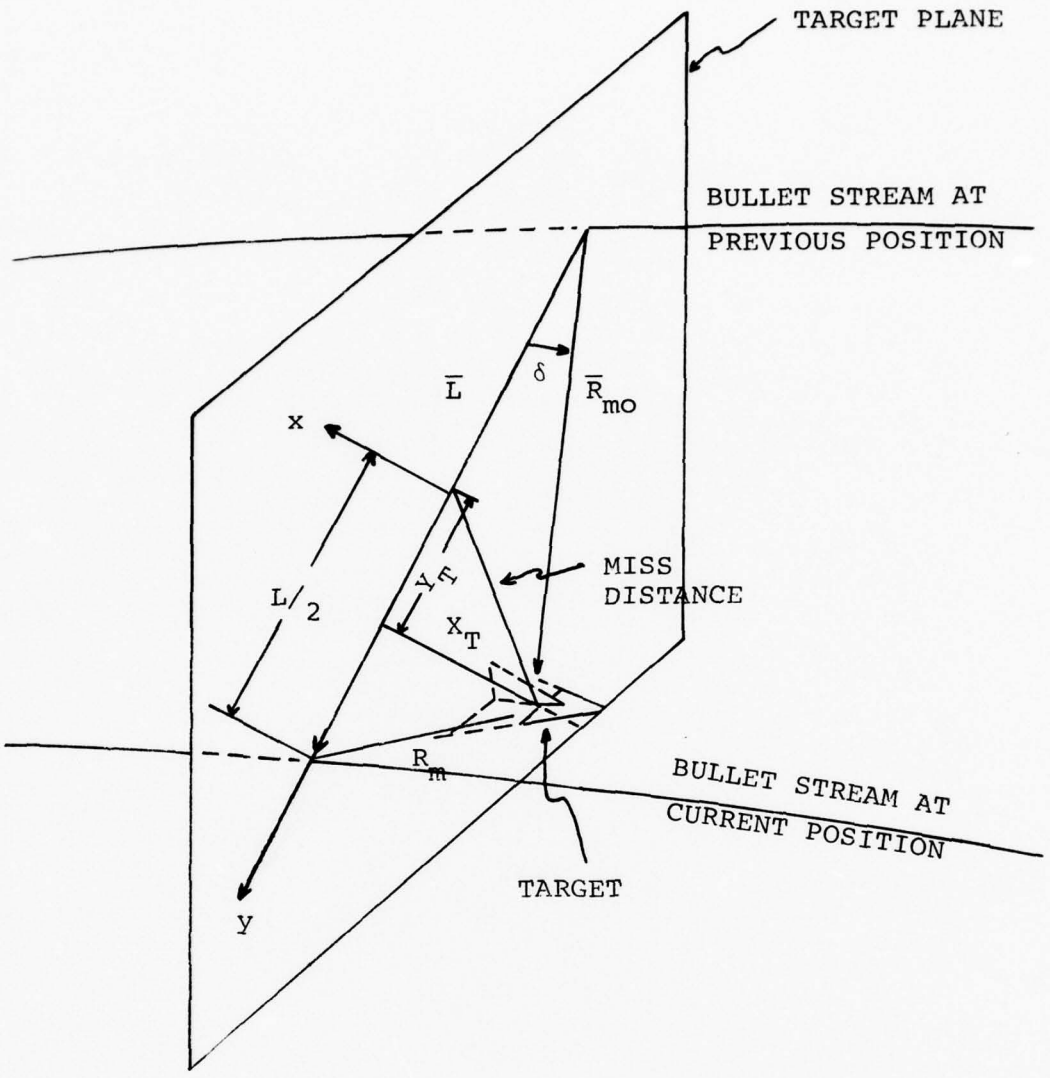


Figure 4. Miss Distance Diagram

This function has the following properties:

$$ERF(-x) = -ERF(x)$$

$$ERF(\infty) = 1/2$$

$$ERF(0) = 0$$

$$ERF(x \geq 3.5) \approx 1/2$$

The error function described in (10) cannot be evaluated in closed form and, therefore, its value must be found numerically.

Four methods of approximating values for the error function were considered.

(1) The use of a table of values for the error function at specified argument values.

(2) A series expansion of the error function.

(3) Numerical integration of the Equation 10.

(4) A least square polynomial representation of the error function.

These four methods of approximating values of the error function were compared relative to computer space and time requirements. As a result of this comparison it was concluded that the best method for this subroutine was a least square polynomial fit. A regression technique was used to fit third through sixth degree polynomials to (10) the error function. A third degree polynomial was chosen for the fit with appropriate linear corrections for small and large x. The specific error function approximation is

$$ERF(x) = K_0 + K_1 x + K_2 x^2 + K_3 x^3 \qquad (11)$$

where, for $x \in [0.0, 0.2]$ $\quad K_1 = 0.39629855$

$$K_0 = K_2 = K_3 = 0$$

$x \in (0.2, 3.5)$ $\quad K_0 = -0.01322336$

$$K_1 = 0.49613046$$

$$K_2 = -0.15942666$$

$$K_3 = 0.01698544$$

$x \in (3.5, \infty)$ $\quad K_0 = 0.5$

$$K_1 = K_2 = K_3 = 0$$

The least square fit of linear polynomials for $0 \leq x < 0.2$ and $x \geq 3.5$ was done to improve the fit in these regions. The error function and the corresponding approximated values are plotted in Figure 5. The actual differences between the true values of the error function and the approximated values is more clearly shown in Figure 6. The corresponding percent difference is shown in Figure 7. The improvement in the approximation in the regions $0 \leq x < 0.2$ and $x \geq 3.5$ can be seen in these figures.

3.2.8 Results

Results obtained from the ACE algorithm for selected input parameters are shown in Figure 8. The number of expected hits is plotted versus miss distance in this figure. No empirical data is available at this time for comparison. If there is a discrepancy in the results obtained and the anticipated results, a detailed study of the approximation to the error function should be considered, because the given approximation could lead to rather large errors in the expected number of hits.

3.2.9 Flowcharts

Figure 9 shows the descriptive flow chart for the ACE algorithm and Figure 10 shows the flow chart for the error function.

3.2.10 Input-Output

INPUTS

VARIABLE NAME	UNITS	DESCRIPTION
AZ	Radians	The azimuth position of the computed projectiles at target range with respect to the target
EL	Radians	The elevation position of the computed projectiles at target range with respect to the target

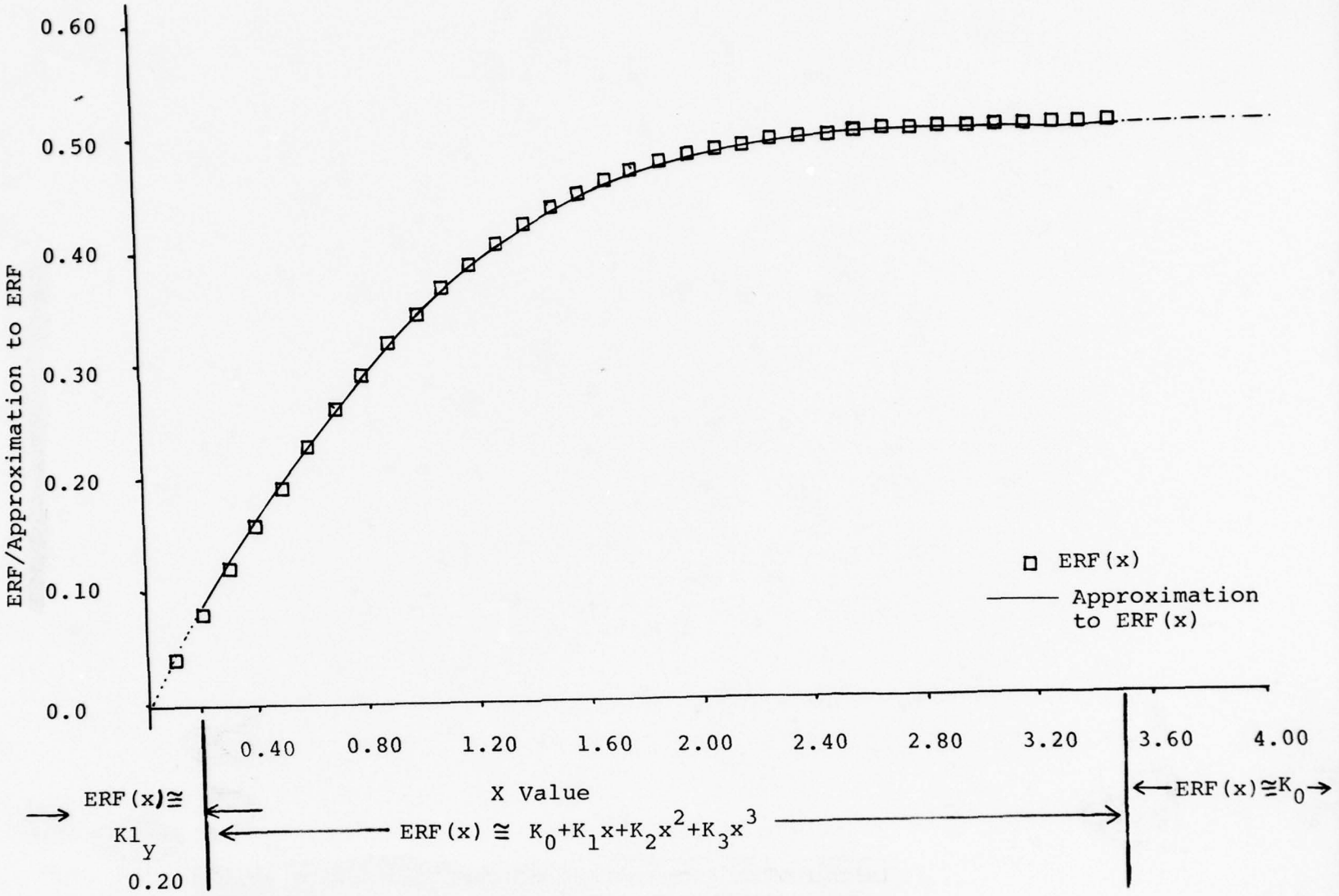


Figure 5. ERF(x) and the Approximation to ERF(x)

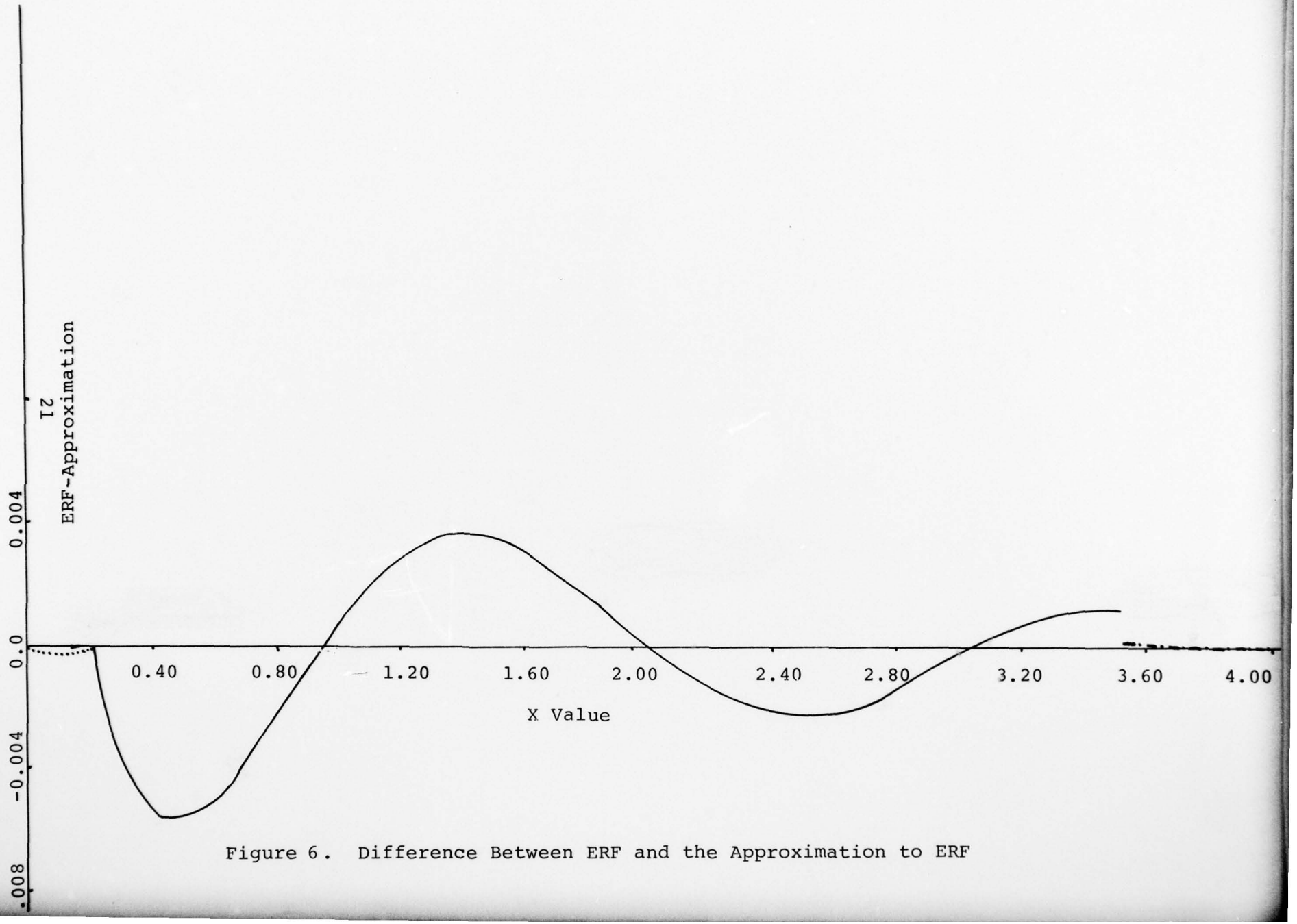


Figure 6. Difference Between ERF and the Approximation to ERF

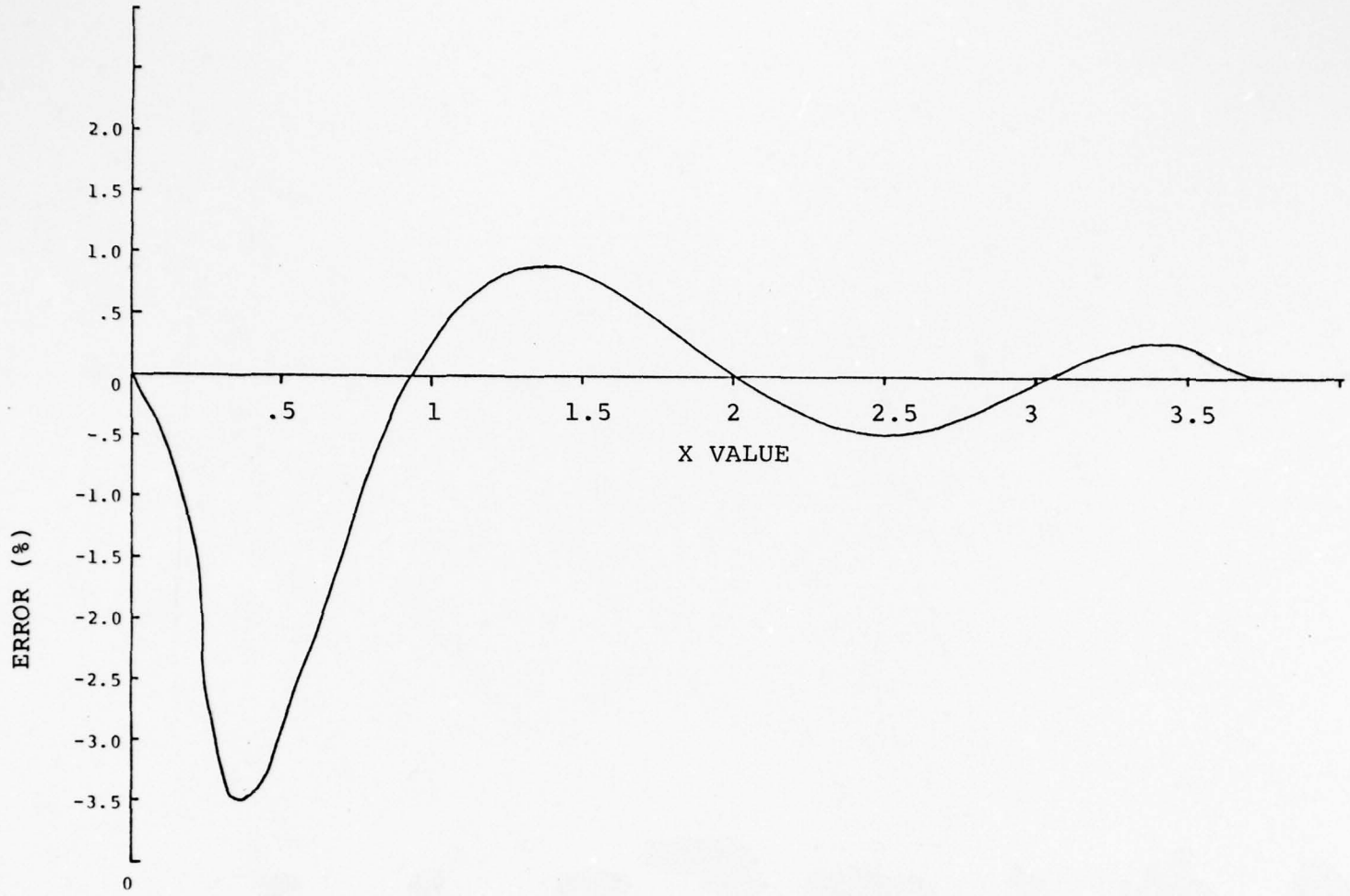


Figure 7. Percent Error Between ERF(x) and the Approximation to ERF(x).

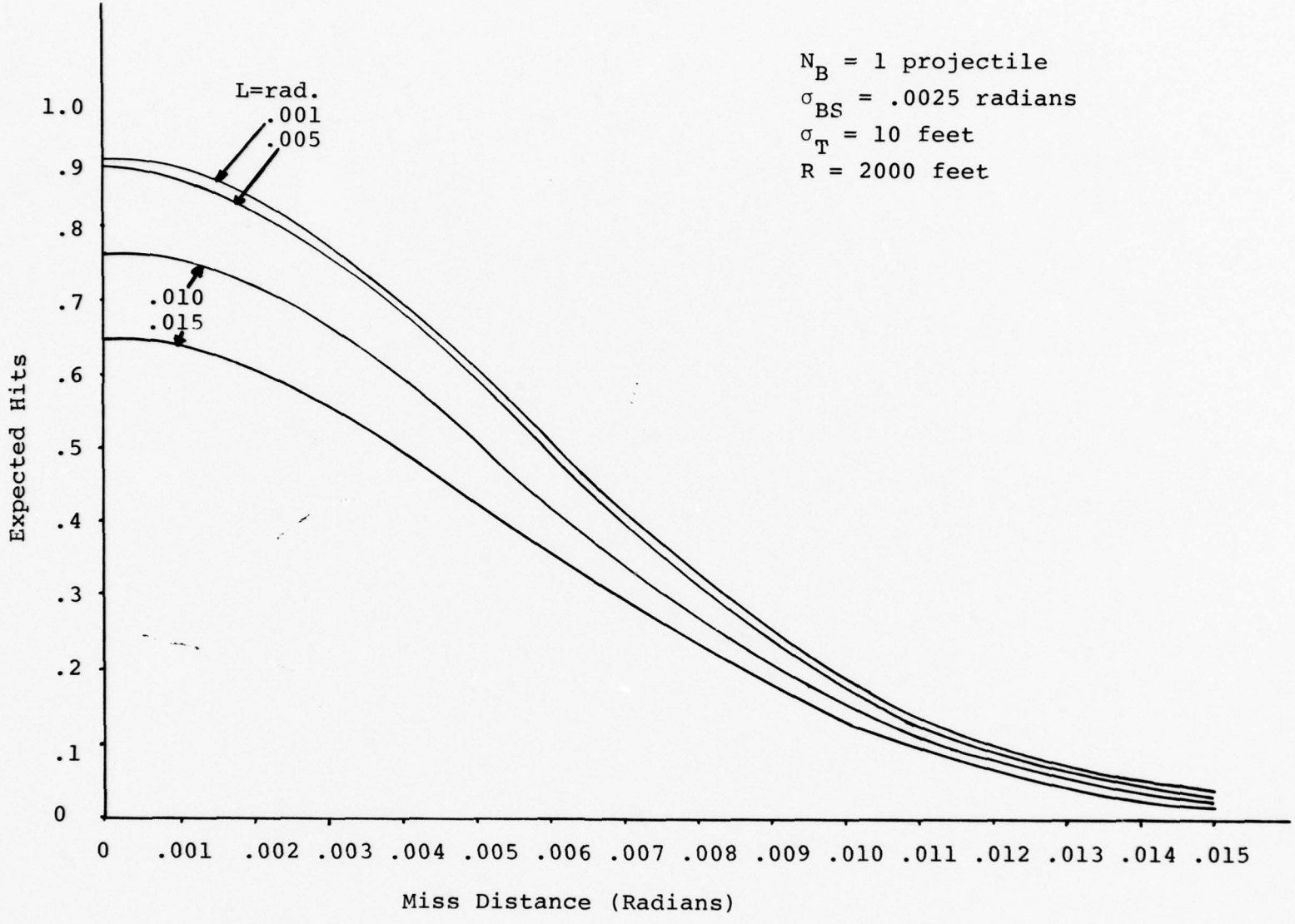
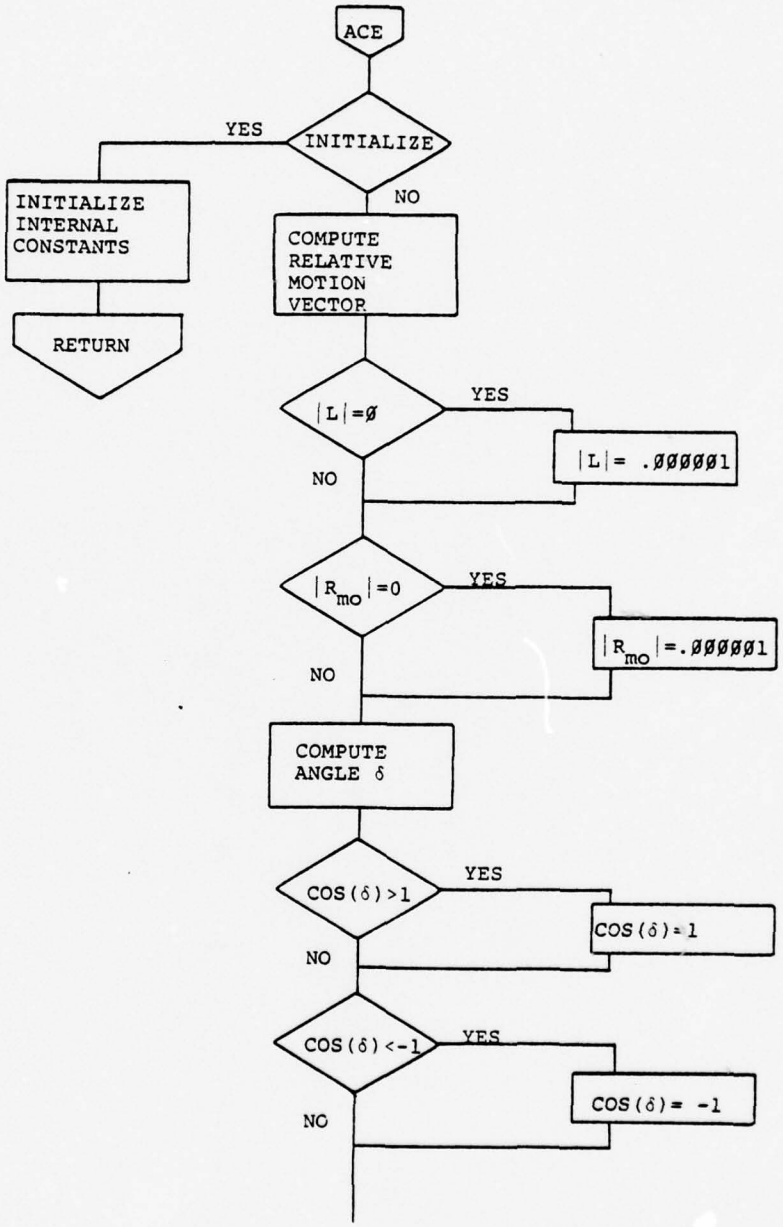


Figure 8. Comparison of Miss Distance and Expected Hits for Several Relative Motion Vectors.

Figure 9. Descriptive Flow Chart



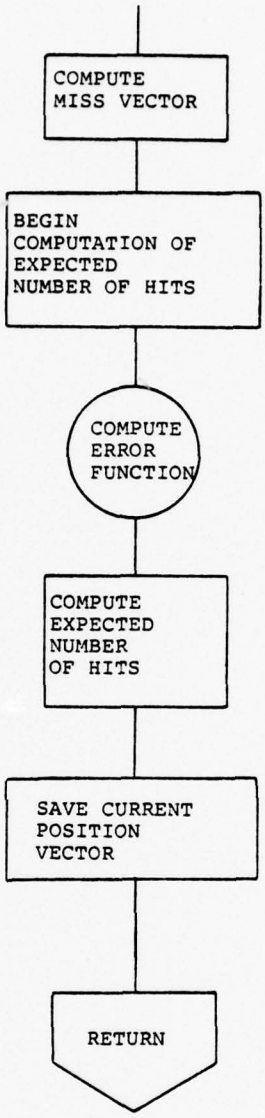
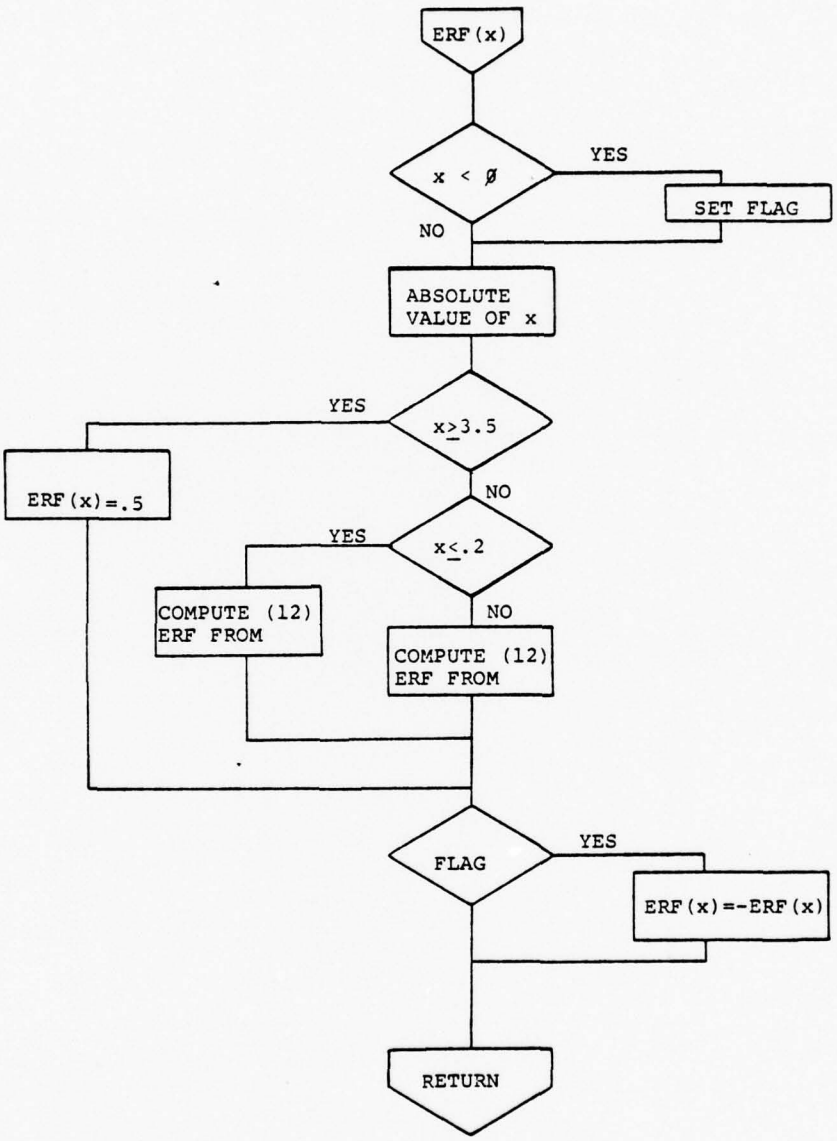


Figure 9. (Continued)

Figure 10. ERF Flow Chart



VARIABLE NAME	UNITS	DESCRIPTION
RANGE	Feet	The range of the target with respect to the attacker
IFLAG	--	The initialization flag that either initializes or runs the subprogram. If IFLAG = 1, then the program is run and computes the expected number of hits. If IFLAG ≠ 1, then the program initializes several constants to be used in the program.

The input variables are in common and the calling program must have a common statement as follows to input values to this subprogram:

```
COMMON/INPT/AZ, EL, RANGE, IFLAG  .
```

OUTPUT

VARIABLE NAME	DESCRIPTION
EXPHTS	This value is the expected number of hits of the projectiles on the target when both are at target range. It is the expected number of hits per computer cycle.

This output variable is in common and the calling program must have a common statement as follows to output values from this subprogram;

```
COMMON/OUTPT/EXPHTS  .
```

INPUT CONSTANTS

VARIABLE NAME	UNITS	DESCRIPTION
BULN	--	The number of projectiles at target range per computer cycle time (fire rate x cycle time)
SIGT	Feet	The standard deviation of the target
SIGBS	Radians	The standard deviation of the projectile stream or projectile dispersion.

The constant variables are common, and the calling program must have a common statement as follows to input these constant values to the subprogram:

```
COMMON/CONST/BULN, SIGT, SIGBS  .
```

INTERNAL CONSTANTS

VARIABLE NAME	UNITS	DESCRIPTION
SQ2PI	--	Square root of 2 x π
E		Exponential
AZOLD	Radians	Previous value of the azimuth position of the computed projectiles at target range with respect to the target
ELOLD	Radians	Previous value of the elevation position of the computed projectile at target range with respect to the target
W2	Radians^2	SIGBS^2

These variables are common only to this subprogram. The later three are in common and are assigned their values at the initialization stage of the subprogram. The common is as follows:

```
COMMON/ACECOM/SQ2PI, E, AZOLD, ELOLD, W2  .
```

The first several values are constant at all times and therefore are in data statements as follows:

```
DATA  SQ2PI/2.5066283/  ,

DATA  E/2.7182818/  .
```

3.2.11 Algorithm Listing

The listing of the ACE algorithm as a coded FORTRAN subroutine is contained in Appendix C.

3.2.12 References

Edwards, Verlan E. and Ted G. Johnson, Air-to-Air Gunfire Control System Evaluator, Honeywell, Inc., Technical Report AFAL-TR-73-20, November 1972.

3.3 FIRE CONTROL DOCUMENTATION

3.3.1 Task Definition

The Large Amplitude Aerospace Research Simulator (LAMARS) facility, which is supported by the AFFDL at WPAFB, is designed to simulate the total environment of an aircraft in flight. The LAMARS facility consists of a cockpit mounted on a large movable arm, various television picture projectors and a computer controlled servo-system. The cockpit area can be modified to meet the cockpit specifications of the aircraft being simulated.

Aircraft motion is simulated by a combination of arm movement and motion picture projection of terrain. The movement of the arm is activated through the controls in the cockpit as the pilot "flys" the aircraft. The terrain over which the pilot is flying is simulated by projecting televised pictures of a terrain onto a screen that surrounds the cockpit area. When the facility is being used to simulate fighter aircraft in combat, the image of a target is projected onto the screen. The flight path of the target aircraft is input to the system.

The LAMARS facility was modified to simulate the F-106 fighter aircraft. The major aircraft responses and terrain along with the target were provided by AFFDL and the fire control algorithms were provided by AFAL. The integration of the AFAL algorithms took place on the ROLM 16/64 computer on the computer deck at LAMARS facility. The ROLM 16/64 computer was used because this computer is part of the air-borne computer system in the F-106 aircraft. The ROLM 16/64 used in the LAMARS facility, is connected to several other computers on the computer deck by use of a Direct Memory Access (DMA) channel. The inputs to the fire control algorithms are passed from the aircraft simulation computer through the DMA to the ROLM 16/64. The outputs of the fire control algorithms are then passed from the ROLM 16/64 through the DMA to the symbol generator

computer to input the symbology to the Heads Up Display for viewing by the pilot.

The addition of the fire control algorithms to the simulation facility allows the pilot to use the symbology on the HUD. The pilot can, as in an actual flight, select what is to be displayed on the HUD and then use the symbology to track the image of the target on the screen. This addition to the LAMARS facility adds to the realism of flight to the pilot and helps to train the pilot to effectively fly a fighter aircraft to track a target while in flight.

3.3.2 Basic Support Efforts Completed

The role of the University of Dayton Research Institute in the integration of the fire control algorithms was basically one of a supportive nature. The AFAL planned the entire integration package using the most common top-down programming approach. The UDRI assisted in integrating five of the fire control algorithms into the planned integration set by the AFAL. Each algorithm, coded in FORTRAN for the ROLM 16/64 computer, had to be adapted for the integration. The adaptation of each algorithm required the specification of the inputs, outputs, and constants by the appropriate common blocks as set by AFAL.

The completed algorithms for the integration are listed in Appendix D. These listings are the compiled versions of the five algorithms adapted for the LAMARS integration.

3.3.3 Documentation Support

Sample documentation was provided to Captain Silverthorn to assist in documenting Hot Line Gunsight (HLGS) and Lead Computing Optical Sight (LCOS).

APPENDIX A
VERIFICATION PROGRAMS

Five assembly language programs were obtained from Reference 1 (see Paragraph 3.1.5 of this volume) to verify the operation of the Cross Assembler. These programs were written for the MDSC computer and compiled by the PDP version of the Cross Assembler on the PDP-11/45 at WPAFB. Verification Programs 1 and 2 are those taken from Reference 1. Verification Programs 3, 4, and 5 are those supplied by the AFAL branch which deals with the MDSC computer. Verification Program 3 has the complete test of the assembly language instruction set and all modes of each instruction. It should be noted that Verification Program 4 has an error indicated. Also, errors have been detected in the verification listing. Consequently, there is a question as to the validity of this program. Verification Programs 1, 2, and 5 are typical of the programs used in the MDSC computer. Verification Programs 1, 2, 3, and 5 have been checked out with the verification listings.

The output format consists of 4 columns.

Column 1 is the card or line number of the assembly language instruction

Column 2 is the location counter (octal)

Column 3 is the machine code (octal)

Column 4 is the input assembly language instruction which generated the machine code

A listing of each of the five verification programs follows.

```
                **MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION**

                                **END OF PASS ONE**

NO ERRORS

**SYMBOL VALUES:

  4 X1ADDR  /    446  1        DAT     /    124  1        DATADD  /    122  1        DADD    /    312  1
  4 DADDR   /    451  1        XADDR   /    512  1        ADD     /    217  1        IDLELOOP/    107  1
  4 RESTOR  /    440  1        SENDOUT /    404  1        SININST /    514  1        CLEAR   /    116  1
  4 CLRCNT  /    123  1        CLRLOOP /     10  1        FMODE   /    107  1        IMSKSV  /     14  1
  4 RMOVINST/    447  1        INT15HZ /      0  1        INT60HZ /      3  1        INITLP  /     15  1
  4 INSTRC  /   1523  1        INSTRUC /   1520  1        SNDOUT  /     50  1        EOF     /     57  1
  4 SOUTINST/    450  1        FPUP    /    105  1        TREND   /     63  1        TRLP    /     24  1
  4 TRLP1   /     42  1        TRLP2   /     51  1        WRDREAD /    452  1        DSCADD  /    120  1
  4 DSTK1   /     60  1        DSCLOOP /     72  1        DSPLOCR /     66  1        DSCTAB  /    117  1
  4 DSPVAL  /     62  1        NSWDS   /     56  1        DTAB    /    347  1        STKADD  /    121  1
  4 STINST  /    513  1        BUF1    /     71  1        NWDS    /     74  1        NWDSP   /     72  1
  1 PWRUP   /      0  1
FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED
```

```
**MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION**

                   **********************************
                   *                                *
                   *         POWER - UP ROUTINE     *
                   *                                *
                   **********************************
       0           PWRUP    EQU      $
  0     3403                IMSK     3              SET SYSTEM MASK
  1    34017                LI       A0,15
  2    60014                ST       A0,12
  3   170320                POUT     A0,0           MASK OUT ALL INT'S
  4   137114                LDR      SP,STKADD      INITIALIZE STACK POINTER
  5    36047                LI       X1,7           SAVE LOC 6
  6    34000                LI       A0,0
  7   136513                LDR      X2,CLRCNT
 10   120000       CLRLOOP  ST       A0,0,X1        SET RAM
 11    32001                ADI      X1,1           TO ZEROES
 12    76775                BINC     X2,CLRLOOP
 13    35305                LI       A2,DAT-ADD      A2 - COUNTER
 14   136105                LDR      X1,DATADD       X1 - TABLE POINTER
                   *
                   *         SET UP PAGE ZERO CONSTANTS
                   *
 15   124000       INITLP   LD       A0,0,X1         GET INITAIL VALUE
 16   126473                LD       X2,ADD-DAT,X1   STORE
 17   160000                ST       A0,0,X2
 20    32001                ADI      X1,1
 21    75373                BINC     A2,INITLP
                   *
                   *        TRANSFER DISPLAY LIST, STACK, CONSTANTS
                   *        BUF1,NWDS,NSWDS,DSTK1
                   *
 22    66066                LD       X1,DSPLOCR      X1-ROM LOC OF DISPLAY LIST
 23    64471                LD       A1,BUF1         A1 - RAM DESTINATION LOC OF DISPLIST
 24   124000       TRLP     LD       A0,0,X1         GET WORD
 25    54057                CMPR     A0,EOF          EOF
 26     4403                BZ       TRLP1           YES
 27   170161                PUSH     A1,A0           INSERT INTO RAM LOC
 30    32001                ADI      X1,1            INC COUNTER
 31     5772                JMP      TRLP
 32   171404       TRLP1    RMOV     A3,X1
 33    41466                SUB      A3,DSPLOCR
 34    31401                ADI      A3,1            A3-LENGTH OF DISPLAY LIST
 35   171543                RNEG     A3,A3           GET 2'S COMPLEMENT
 36    61474                ST       A3,NWDS         NWDS-TEMP LOC FOR DISP GENERATOR
 37    61472                ST       A3,NWDSP        NWDSP-LENGTH OF DISP LIST (2'S COMP.)
 40    64066                LD       A0,DSPLOCR
 41    64471                LD       A1,BUF1
 42   170041                RSUB      A0,A1          A0-RELOCATION OFFSET OF DISP LIST
 43    60005                ST       A0,5             TEMP LOC.
 44    64071                LD       A0,BUF1
 45    32001                ADI      X1,1            X1-START LCO OF DISP LIST POINTER TAB
 46    62064                ST       X1,X'34'
 47    65060                LD       A2,DSTK1        A2-RAM DESTINATION OF D.L. TABLE
 50    65462                LD       A3,DSPVAL       A3-RAM LOC OF D.L. UPDATE VALUES
 51   124000       TRLP2    LD       A0,0,X1         GET WORD
 52    54057                CMPR     A0,EOF          EOF
 53     4407                BZ       TREND           YES
 54    40005                SUB      A0,5            RELOCATION OFFSET FOR POINTER
 55   172400                RMOV     X2,A0
 56   164400                LD       A1,0,X2         CURRENT CONTENTS (ROM) POINTED TO
 57   170162                PUSH     A2,A0           STORE POINTER
 60   170563                PUSH     A3,A1           STORE VALUE IN UPDATE TABLE
```

```
 63      61    52001            ADI       X1,1
 64      62     5766            JMP       TRLP2
 65      63   171404  TREND     RMOV      A3,X1
 66      64    41464            SUB       A3,X'34'
 67      65    61456            ST        A3,NSWDS
 68      66     5400            NOP
 69                    *
 70                    *        INITIALIZE SCAN CONVERTER
 71                    *
 72      67   136027            LDR       X1,DSCTAB
 73      70   136427            LDR       X2,DSCADD
 74      71    35743            LI        A3,DADD-DTAB
 75      72   124000  DSCLOOP   LD        A0,0,X1
 76      73   164400            LD        A1,2,X2
 77      74       50            JSRZ      *SNDOUT
 78      75    35356            LI        A2,-10
 79      76     5400            NOP                          DELAY
 80      77     5420            NOP
 81     100    73375            BINC      A2,$-2
 82     101    52001            ADI       X1,1
 83     102    32401            ADI       X2,1
 84     103    75766            BINC      A3,DSCLOOP
 85                    *
 86                    *        INITIAL PASS THRU INT50HZ
 87                    *
 88     104        0            JSRZ      *INT15HZ
 89     105    34000            LI        A0,0
 90     106    60105            ST        A0,FPUP           CLEAR POWER-UP FLAG
 91                    *
 92                    *
 93     107     5400  IDLELOOP  NOP
 94     110     5410            IMSK      8
 95     111    34000            LI        A0,0
 96     112    60010            ST        A0,12
 97     113   170020            POUT      A0,0              RE-ENABLE INTERRUPTS
 98     114    64116            LD        A0,CLEAR
 99     115     5371            BNZ       IDLELOOP
100     116     5661            JMP       PWRUP             CLEAR FLAG SET
101                   *
102                   *        DATA
103                   *
104          50       SNDOUT    EQU       X'28'
105           0       INT15HZ   EQU       0
106           3       INT60HZ   EQU       X'3'
107          60       DSTK1     EQU       X'30'
108          62       DSPVAL    EQU       X'32'
109          57       EOF       EQU       X'2F'
110          66       DSPLOCR   EQU       X'36'
111          71       RUF1      EQU       X'39'
112          56       NSWDS     EQU       X'2E'
113          74       NWDS      EQU       X'3C'
114          72       NWDSP     EQU       X'3A'
115         105       FPUP      EQU       X'45'
116         116       CLEAR     EQU       X'4E'
117     117     347   DSCTAB    DATA      DTAB
118     120     312   DSCADD    DATA      DADD
119     121    1000   STKADD    DATA      1536
120     122     124   DATADD    DATA      DAT
121     123  171000   CLMENT    DATA      -X'600'
122                   *        INITIAL DATA VALUES
123         124       DAT       EQU       $
124     124   10540             DATA      X'1160'           INT15HZ
125     125   15140             DATA      X'1A60'           S.P.I.
126     126   15665             DATA      X'1B85'           S.T.I.
127     127   15550             DATA      X'1A30'           50 HZ
128     130     520             DATA      X'150'            PPIANGS
129     131   10660             DATA      X'1180'           SWCHRD
```

```
130   132   11620          DATA    X'1390'     SNCHART
131   133   10117          DATA    X'1C4F'     PHCX0
132   134   10107          DATA    X'1C47'     MA1PHC
133   135   10052          DATA    X'112A'     WRDRD
134   136   10076          DATA    X'1C3E'     PHCMA1
135   137      0           DATA    0            SPSYM
136   140   1000           DATA    512          CURRENT DISPLAY BUFFER
137   141   13130          DATA    X'1658'     DISPLAYS
138   142      1           DATA    1            FMODE
139   143   10004          DATA    X'1184'     SNDOUT
140   144   15240          DATA    X'1CA0'     MUXIN
141   145      1           DATA    1            FPUP
142   146    400           DATA    256          DSTX1
143   147   16270          DATA    X'1CB8'     CORDIC
144   150    600           DATA    384          DSPVAL
145   151   14002          DATA    X'1902'     DISPLIST
146   152   1000           DATA    512         BUF 1 ADDRESS
147   153   177777         DATA    X'FFFF'      EOF
148   154   16410          DATA    X'1008'     PPIPTR
149   155   177777         DATA    -1          60 HZ RESET
150   156   10000          DATA     4096       PWRUP
151   157      170         DATA    120           ELVID
152   160      1           DATA    1           ERASE ENABLE
153   161      0           DATA    0           F50HZ
154   162      1           DATA    1           FPPI
155   163      0           DATA    0           FBIT
156   164      0           DATA    0           TV
157   165      0           DATA    0           MASK40
158   166      0           DATA    0           AGECNTR
159   167      1           DATA    1           MCIMM
160   170      0           DATA    0            GZLCNTR
161   171      0           DATA    0            GZLDIR
162   172      0           DATA    0           AGEDLAY
163   173   15652          DATA    X'18AA'     XFRENBL
164   174   17400          DATA    X'1F00'      BIT2
165   175   17640          DATA    X'1FA0'     BIT1
166   176      0           DATA    0            SCMODE
167   177      0           DATA    0           DSMODE
168   200   174120         DATA    X'F850'      15HZ
169   201   174120         DATA    X'F850'      SPI
170   202   174120         DATA    X'F850'      STI
171   203   174120         DATA    X'F850'     60HZ
172   204     1240         DATA    X'2A0'       TEM MASK
173   205   174120         DATA    X'F850'      BIT DUMMY
174   206     5400         NOP                  SNDOUT
175   207     5400         NOP
176   210   174120         RTS
177   211     5400         NOP                 WRDRD
178   212     5400         NOP
179   213   174120         RTS
180   214     1526         DATA    X'356'      WTIMER-ADDRESS
181   215   170360         IOCP    0           WTIMER
182   216   174120         RTS
183                  *      INITIAL DATA ADDRESSES
184   217      0     ADD     DATA    0
185   220      1             DATA    1
186   221      2             DATA    2
187   222      3             DATA    3
188   223     34             DATA    X'1C'       PPIABGS
189   224     41             DATA    X'21'       SNCHRC
190   225     46             DATA    X'26'        SNCHART
191   226     52             DATA    X'2A'       PHCSCL
192   227     53             DATA    X'2B'       MA1PHC
193   230     56             DATA    X'2C'        WRDRD
194   231     55             DATA    X'2D'       PHCMA1
195   232     67             DATA    X'37'        SPSYM
196   233     73             DATA    X'3B'        CURRENT DISPLAY BUFFER
```

```
197   234    42            DATA    X'22'       DISPLAYS
198   235   107            DATA    X'47'        FMODE
199   236    50            DATA    X'28'       SNDOUT
200   237    51            DATA    X'29'       MUXIN
201   240   105            DATA    X'45'        FPUP
202   241    60            DATA    X'30'       USTAI
203   242    61            DATA    X'31'        CORDIC
204   243    62            DATA    X'32'        DSPVAL
205   244    56            DATA    X'36'        DISPLOCR
206   245    71            DATA    X'39'       BUFI  ADDRESS
207   246    57            DATA    X'2F'        EOF
208   247   161            DATA    X'71'       PPIPTR
209   250   317            DATA    X'CF'       60HZ RESET
210   251    37            DATA    X'1F'       PWRUP
211   252   157            DATA    X'6F'       ELV10
212   253    40            DATA    X'20'       ERASE ENABLE
213   254   102            DATA    X'42'       F30HZ
214   255    77            DATA    X'3F'       FPPI
215   256   110            DATA    X'48'       FBIT
216   257   117            DATA    X'4F'       TV
217   260   171            DATA    X'79'       MASK40
218   261   177            DATA    X'7F'       AGENCNTR
219   262   265            DATA    X'B5'        MCIMM
220   263   154            DATA    X'6C'        GZLCNTR
221   264   172            DATA    X'7A'        GZLDIR
222   265   207            DATA    X'87'        AGEDLAY
223   266   210            DATA    X'88'       XFRNBL
224   267   376            DATA    X'FE'        BIT2
225   270   377            DATA    X'FF'        BIT1
226   271   111            DATA    X'49'        SCMODE
227   272   112            DATA    X'4A'        DSMODE
228   273  5000            DATA    X'A00'       15HZ
229   274  5100            DATA    X'A40'       SPI
230   275  5200            DATA    X'A80'       STI
231   276  5300            DATA    X'AC0'       60HZ
232   277    16            DATA    X'E'
233   300  3120            DATA    X'650'       DUMMY
234   301  1520            DATA    X'350'       SNDOUT
235   302  1521            DATA    X'351'
236   303  1522            DATA    X'352'
237   304  1523            DATA    X'353'       WBORD
238   305  1524            DATA    X'354'
239   306  1525            DATA    X'355'
240   307   314            DATA    X'CC'       WTIMER
241   310  1526            DATA    X'356'      WTIMEH
242   311  1527            DATA    X'357'
243                 *
244                 *       INITAIL MODULE ADDRESSESS
245                 *
246   312     0     DADD    DATA    0
247   313     1             DATA    1
248   314     2             DATA    2
249   315     3             DATA    3
250   316     4             DATA    4
251   317     5             DATA    5
252   320     6             DATA    6
253   321     7             DATA    7
254   322    22             DATA    16
255   323    30             DATA    24
256   324    31             DATA    25
257   325    55             DATA    27
258   326    34             DATA    28
259   327    35             DATA    29
260   330    36             DATA    30
261   331    37             DATA    31
```

```
262   332      40                DATA    32
263   333      41                DATA    33
264   334      42                DATA    34
265   335      43                DATA    35
266   336      44                DATA    36
267   337      45                DATA    37
268   340      46                DATA    38
269   341      50                DATA    40
270   342      52                DATA    42
271   343      55                DATA    45
272   344      56                DATA    46
273   345      76                DATA    54
274   346      72                DATA    58
275                      *
276                      *       INITIAL MODULE VALUES
277                      *
278   347  176000  DTAB  DATA    X'FC00'         0
279   350  155700        DATA    X'D8C0'          1
280   351  100000        DATA    X'8000'         2
281   352    4000        DATA    X'800'          3
282   353       0        DATA    0               4
283   354   46640        DATA    X'4DA0'         5
284   355       0        DATA    0               6
285   356  170000        DATA    X'F000'         7
286   357       0        DATA    0               16
287   360       0        DATA    X'0000'         24
288   361  100000        DATA    X'8000'         25
289   362  140700        DATA    X'C1C0'         27
290   363       0        DATA    X'000'          28
291   364  100000        DATA    X'8000'         29
292   365   17000        DATA    X'1E00'         30
293   366  100000        DATA    X'8000'         31
294   367   10000        DATA    X'1000'         32
295   370   47760        DATA    X'4FF0'         33
296   371       0        DATA    X'000'          34
297   372   37000        DATA    X'3E00'         35
298   373  151626        DATA    X'0396'         36
299   374       0        DATA    0               37
300   375       0        DATA    0               38
301   376    4000        DATA    X'800'          40
302   377    4700        DATA    X'9C0'          42
303   400  100000        DATA    X'8000'         45
304   401       0        DATA    0               46
305   402  170017        DATA    X'F00F'         56
306   403  170000        DATA    X'F000'         58
307                      *
308                      *       SENDOUT
309                      *       A0 = WORD TO BE SENT OUT
310                      *       A1 = MODULE ADDRESS
311                      *
312        107           FMODE   EQU     X'47'           MODE CHANGE FLAG
313         14           IMSKSV  EQU     X'C'
314   404  170166  SENDOUT PUSH    SP,0
315   405  170566        PUSH    SP,1
316   406  171166        PUSH    SP,2
317   407  171566        PUSH    SP,3
318   410  172166        PUSH    SP,X1
319   411  171002        RMOV    2,0
320   412    4017        LI      0,X'F'
321   413  170320        POUT    0,0             MASK OUT ALL INTERRUPTS
322   414  170002        RMOV    0,2
323   415  136033        LDR     X1,OADDR
324   416  172021        RADD    X1,1
325   417  122000        ST      0,0,X1
326   420  136025        LDR     X1,X1ADDR
327   421   35460        LI      3,X'30'         BUILD RMOV INSTRUC
```

```
                    171461          RAND    3,1
                    175623          SHL     3,4
                    135022          LDR     2,RMOVINST
                    171125          ROR     2,5
                    121000          ST      2,0,X1
                    135020          LDR     2,SOUTINST      BUILD SERIAL OUT INSTRUC
                    171103          ROR     2,3
                     35417          LI      3,X'F'
                    170463          RAND    1,3
                    171101          ROR     2,1
                    121001          ST      2,1,X1
                      2010          JSR     *X1ADDR         JUMP TO RAM TO EXECUTR
                     64014          LD      A0,IMSKSV       RESTORE INTERRUPT MASK
                    170320          PDUT    0,0             MASK INTERRUPTS BACK IN
                    172206  RESTOR  POP     SP,X1
                    171606          POP     SP,3
                    171206          POP     SP,2
                    170606          POP     SP,1
                    170206          POP     SP,0
                    174120          RTS
                      1520  X1ADDR  DATA    INSTRUC
                    170000  RMOVINST DATA   X'F000'         1111 00XX 0000 0000
                    170340  SOUTINST DATA   X'F0E0'         1111 00XX 1110 XXXX
                      2220  QADDR   DATA    X'4440'         RAM ADDRESS TO STORE LAST OUT
          1520      INSTRUC EQU     X'350'          RAM ADDRESS WHERE 'SOUT' EXECUTED
                            *
                            *       XRDREAD
                            *       X0 = MODULE ADDRESS FROM WHICH WORD IS TO BE INPUT
                            *       A1 = MEMORY ADDRESS INTO WHICH WORD IS TO BE STORED
                            *
                    170166  XRDREAD PUSH    SP,0
                    170566          PUSH    SP,1
                    171166          PUSH    SP,2
                    171566          PUSH    SP,3
                    172166          PUSH    SP,X1
                    171000          RMOV    2,0
                     34017          LI      0,X'F'
                    170320          PDUT    0,0             MASK OUT ALL INTERRUPTS
                    170002          RMOV    0,2
                    136026          LDR     X1,XADDR
                     35460          LI      3,X'30'         BUILD STORE INSTRUCTION
                    171460          RAND    3,0
                    175623          SHL     3,4
                    135023          LDR     2,STINST
                    171103          ROR     2,3
                    121001          ST      2,1,X1
                    135021          LDR     2,SININST       BUILD SERIAL IN INSTRUC
                    171103          ROR     2,3
                     35417          LI      3,X'F'
                    170063          RAND    0,3
                    171100          ROR     2,0
                    121000          ST      2,0,X1
                    176001          RMOV    X1,1
                      2010          JSR     *XADDR          JUMP TO RAM TO EXECUTE SIN
                     64014          LD      A0,IMSKSV       RESTORE INTERRUPT MASK
                    170320          PDUT    0,0             MASK INTERRUPTS BACK IN
                    172206          POP     SP,X1
                    171606          POP     SP,3
                    171206          POP     SP,2
                    170606          POP     SP,1
                    170206          POP     SP,0
                    174120          RTS
```

```
390      512     1523   XADDR    DATA     INSTRC
391      513   120000   STINST   DATA     X'A000'          1010 00XX 0000 0000
392      514   174340   SININST  DATA     X'F8E0'          1111 10XX 1110 XXXX
393           1523      INSTRC   EQU      X'353'           RAM ADDRESS WHERE 'SIN' EXECUTED
394                     *
395     17400                    $LOC     X'1F00'          B.I.T.
396  17400   174120              RTS                       DUMMY
397     17640                    $LOC     X'1FA0'          PATCH
398  17640   174120              RTS                       RETURN
399                              END

                                                **END OF PASS TWO**

NO ERRORS

    335 (   517 OCT) WORDS OF CODE GENERATED
```

MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION

END OF PASS ONE

NO ERRORS

**SYMBOL VALUES:

4 X1FF	/ 17565	1	X900	/ 17621	1	X989	/ 17617	1	MA1AZ	/ 362	1
4 FHIT	/ 110	1	X800	/ 17620	1	SCMODE	/ 111	1	XC0	/ 17730	1
4 DELAY	/ 17727	1	SENMOV	/ 17711	1	SENMOV1	/ 17714	1	BIT7	/ 17564	1
4 BITAZC	/ 365	1	HITAZR	/ 370	1	BITCNT	/ 366	1	BITDIR	/ 372	1
4 BITFREEZ	/ 365	1	BITFRM	/ 371	1	BIAS	/ 17614	1	BIAS1	/ 17607	1
4 BIAS2	/ 17611	1	BIAS3	/ 17613	1	BITS0	/ 17415	1	BITS1	/ 17414	1
4 BITS10	/ 17425	1	BITS20	/ 17541	1	BITSET	/ 17400	1	BITVID	/ 373	1
4 BITW046	/ 3	1	BITX3	/ 17643	1	BITX4	/ 17662	1	BITX5	/ 17666	1
4 BITX6	/ 17725	1	BITXEQ	/ 17640	1	VIDEO	/ 33	1	VIDPOT	/ 17562	1
4 CLRCNT	/ 367	1	ANTDIR	/ 153	1	INWRD40	/ 113	1	INWRD41	/ 15	1
4 RNGOFFV	/ 17566	1	SNDOUT	/ 50	1	MODADR	/ 17570	1	MODETABL	/ 17731	1
4 MODECHNG	/ 107	1	WRD01	/ 17677	1	FPUP	/ 105	1	WRD1A	/ 17705	1
4 WRD1B	/ 17703	1	WRD41V	/ 17567	1	WRDRD	/ 54	1	LSTFRM	/ 6	1
4 USC	/ 17572	1	OSC90	/ 17571	1	OSCM	/ 17605	1	OSCV	/ 17606	1
3 STEP	/ 2	1	AZL	/ 57	1	AZR	/ 7	1			

FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED

```
                        ************************************
                        *
                        *       BUILT IN TEST ROUTINES
                        *
                        ************************************
                        SLOC    X'1F00'
                        *       15 HZ SETUP
      17400     BITSET  EQU     $               SET UP (15HZ)
17400  64105            LD      A0,FPUP
17401   5011            BNZ     BITS0           NO BIT ON POWER UP
17402  134161           LDR     A0,BIT7
17403  46015            AND     A0,INWRD41      CHECK FOR BIT MODE
17404   4407            BZ      BITS1
17405  64110            LD      A0,FBIT         NOT BIT,CHECK PREV. FRAME
17406   4404            BZ      BITS0
17407  34001            LI      A0,1            BIT ON PREV. FRAME
17410  60107            ST      A0,MODECHNG     SIGNAL MODE CHANGE
17411  34000            LI      A0,0
17412  60110            ST      A0,FBIT
17413  174120   BITS0   RTS
                        *
                        *       BIT MODE SELECTED
                        *
17414  64110    BITS1   LD      A0,FBIT         BITMODE
17415   5007            BNZ     BITS10          FIRST PASS
17416  60366            ST      A0,BITCNT       YES, INITIALIZE COUNTERS
17417  60371            ST      A0,BITFRM
17420  60372            ST      A0,BITDIR       SET RIGHT TO LEFT DIRECTION
                        *                       INITIALIZE SENSOR SCAN
17421  34007            LI      A0,AZR
17422  60107            ST      A0,MODECHNG     SIGNAL MODE CHANGE
17423  60363            ST      A0,BITAZC
17424  60110            ST      A0,FBIT
                        *       SET UP VARIABLES: MUXIN WON'T READ WITH FBIT SET
      17425     BITS10  EQU     $
                        *                       HORIZON
17425  64766            LD      A1,BITCNT
17426  174620           SHL     A1,1            X2 FREQ
17427   6142            JSR     OSC
17430  174226           SHL     A0,7            SCALE TO 7FFF, 1/4 RANGE
17431  60124            ST      A0,X'54'        CPHI
17432  35377            LI      A2,-1
17433  171220           SHR     A2,1            A2,7FFF
17434  170022           RADD    A0,A2
17435  60123            ST      A0,X'53'        SPHI
17436  34001            LI      A0,1
17437  174236           SHL     A0,15           A0=X'8000'
17440  60121            ST      A0,X'51'        STHLTA
17441  64766            LD      A1,BITCNT
17442   6127            JSR     OSC
17443   6143            JSR     BIAS1           CTR=989
17444  60120            ST      A0,X'50'        YTRIM
                        *                       X POSITIONS
17445  64766            LD      A1,BITCNT
17446  174621           SHL     A1,2            X4 FREQ
17447   6122            JSR     OSC             RANGE=3F
17450   6136            JSR     BIAS1           CTR=989
17451  60134            ST      A0,X'5C'        X DOT
17452  174621           SHL     A1,2            RANGE=FF
17453   6133            JSR     BIAS1           CTR=989
17454  60132            ST      A0,X'5A'        MARK X
17455  60133            ST      A0,X'5B'        XLINE
                        *                       MARKY,INLEFT,YFIRE
```

```
 64   17456   64766          LD      A1,BITCNT
 65   17457   174621         SHL     A1,2            X4 FREQ
 66   17460     6110         JSR     OSC90
 67   17461     6127         JSR     BIAS2           CTR=800
 68   17462   60152          ST      A0,X'5A'        MARKY
 69   17463   60137          ST      A0,X'5F'        IHLEFT
 70   17464   60140          ST      A0,X'60'        YFIRE
 71                *                                 AMARKEL,ELVID,TGTEL
 72   17465   64772          LD      A1,SITDIR
 73   17466   174620         SHL     A1,5
 74   17467     6117         JSR     BIAS1           CTR=989
 75   17470   60141          ST      A0,X'61'        AMARKEL
 76   17471   60012          ST      A0,X'A'         ELVID TEMP
 77   17472   30417          ADI     A1,X'F'
 78   17473     6113         JSR     BIAS1           CTR=989
 79   17474   60126          ST      A0,X'56'        TGTEL
 80                *                                 YDOT
 81   17475   64766          LD      A1,BITCNT
 82   17476   174623         SHL     A1,4            X16 FREQ
 83   17477     6072         JSR     OSC
 84   17500     6106         JSR     BIAS1           CTR989
 85   17501   60145          ST      A0,X'50'        YDOT
 86                *                                 XREF,YREF
 87   17502   134114         LDR     A2,X989
 88   17503   60142          ST      A0,X'62'        XREF
 89   17504   60143          ST      A0,X'63'        YREF
 90                *                                 TGT CIRCLE
 91   17505   64766          LD      A1,BITCNT
 92   17506   174622         SHL     A1,3            X8 FREQ
 93   17507   134055         LDR     A0,X1FF
 94   17510   170460         RAND    A1,A0
 95   17511     6101         JSR     BIAS3
 96   17512   60127          ST      A0,X'57'        TGTDIAM
 97   17513   54000          LI      A1,0
 98   17514     6072         JSR     BIAS1
 99   17515   60130          ST      A0,X'58'        TGTGAP
100                *                                 RANGE OFFSET
101   17516   134047         LDR     A0,RNGOFFV
102   17517   60222          ST      A0,X'82'        RANGE OFFSET
103                *
104                *                                 SET BIT FREEZE FLAG
105   17520   54010          LI      A0,8
106   17521   46015          AND     A0,INWRD41
107   17522   60365          ST      A0,BITFREEZ
108                *
109                *                                 SET-UP-BIT-SWITCH PATTERN
110   17523   52000          LI      A2,0
111   17524   54004          LI      A0,4            CHECK FOR CLEAR
112   17525   46015          AND     A0,INWRD41
113   17526     5012         BNZ     BITS20
114   17527   64367          LD      A2,CLRCNT       WAIT FOR STEADY SIGNAL
115   17530   31001          ADI     A2,1            INCREMENT COUNT
116   17531   71002          CPI     A2,2
117   17532     5006         BNZ     BITS20
118   17533   64371          LD      A0,BITFRM       CLEAR SWITCH ON
119   17534   30001          ADI     A0,1            FRAME=FRAME+1
120   17535   70006          CPI     A0,LSTFRM       CHECK FOR LAST FRAME
121   17536     7001         BN      $+2
122   17537   30000          LI      A0,0
123   17540   60371          ST      A0,BITFRM
```

```
124          17541          BITS20   EQU      $                SET SWITCH SELECTION
125    17541   61567                 ST       A2,CLRCNT
126    17542   64371                 LD       A0,BITFRM
127    17543   130024                LDR      A1,MODADR
128    17544   172020                MADD     X1,A0
129    17545   124400                LD       A1,0,X1
130    17546    60513                ST       A1,INWRD40
131    17547    34565                LI       A1,X'75'         PPI+VIDEO+ERASE SEL,
132    17550    46415                AND      A1,INWRD41
133    17551   134015                LDR      A0,WRD41V
134    17552   170500                MOR      A1,A0
135    17553    60415                ST       A1,INWRD41
136                                  *                         READ RADAR VIDEO POT
137    17554   135005                LDR      A2,VIDPOT
138    17555   171552                SOUT     A2,X'A'
139    17556    35347                LI       A2,-25
140    17557    75377                BINC     A2,$
141    17560   175552                SIN      A2,X'A'
142    17561    61033                ST       A2,VIDEO
143    17562     4700  VIDPOT   DATA   X'9C0'          CHANNEL X'13'
144              33    VIDEO    EQU      X'1B'           VIDEO GAIN POT VALUE
145    17563   174120                RTS                      EXIT
146                                  *
147                                  *        DEFINITIONS
148                                  *
149             113    INWRD40  EQU      X'4B'
150              15    INWRD41  EQU      X'20'
151             110    FAIT     EQU      X'48'
152             366    BITCNT   EQU      X'F6'
153             363    BITAZC   EQU      X'F3'
154             372    BITDIR   EQU      X'FA'
155             365    BITFREEZ EQU      X'F5'
156             371    BITFRM   EQU      X'F9'
157             373    BITVID   EQU      X'FB'
158             367    CLRCNT   EQU      X'F7'
159              54    WRDRD    EQU      X'2C'
160    17564     200  BIT7     DATA     X'0080'
161             107    MODECHNG EQU      X'47'           FMODE
162             105    FPUP     EQU      X'45'
163             111    SCMODE   EQU      X'49'
164              50    SNDOUT   EQU      X'28'
165             153    ANTDIR   EQU      X'6B'
166             370    BITAZR   EQU      X'F8'
167             362    MAIAZ    EQU      X'F2'
168              67    AZL      EQU      55              LEFT LIMIT
169               7    AZR      EQU      7
170               3    BITWD46  EQU      3               BIT MODE AND AZ SELECT FOR WD 46
171               2    STEP     EQU      2               AZ SCAN STEP
172    17565     777  X1FF     DATA     X'1FF'
173    17566   50000  RNGOFFV  DATA     X'5000'
174    17567   141414 WRD41V   DATA     X'C38C'
175    17570   17731  MODADR   DATA     MODETABL        MODE TABLE ADDRESS
176                                  *
177                                  * TRIANGLE GENERATOR
178                                  *
179                                  *   A1=INPUT COUNTER
180                                  *   A0=OUTPUT OSC VALUE, RANGE -40 TO 3F
181    17571   30500  OSC90    ADI      A1,X'40'        90 DEG. PHASE SHIFT
182    17572   134012 OSC      LDR      A0,OSCM
183    17573   170061          MAND     A0,A1           VAL = 8 MSB OF COUNT
184    17574   135011          LDR      A2,OSCV
185    17575   170122          RCMP     A0,A2
186    17576     7003          BN       3+4             VAL .LT. FULL SCALE, A0=VAL.
187    17577   170140          RNEG     A0,A0
188    17600   170022          MADD     A0,A2
189    17601   170022          MADD     A0,A2           VAL .GE. FULL SCALE, A0=2*F.S. - VAL
190    17602   30300           ADI      A0,-X'40'       SHIFT TO CTR ON 0
```

```
191   17604   170400          RMOV    A1,A0
192   17605   174120          RTS
193   17606      177  OSCM    DATA    X'7F'       2xFULL SCALE MAGN
194   17607      200  OSCV    DATA    X'80'       FULL SCALE
195                     *
196                     * DISPLAY VARIABLE BIAS
197                     *
198   17610   134007  BIAS1   LDR     A0,X989
199   17611     5003          JMP     BIAS
200   17612   134006  BIAS2   LDR     A0,X800
201   17613     5001          JMP     BIAS
202   17614   134005  BIAS3   LDR     A0,X900
203   17615   170021  BIAS    RADD    A0,A1
204   17616   174223          SHL     A0,4        SCALE FOR OUTPUT
205   17617   174120          RTS
206   17620     4671  X989    DATA    X'989'
207   17621     5400  X800    DATA    X'800'
208   17622     4400  X900    DATA    X'900'
```

```
                        SLOC    X'1FA0'
                *
                *       BIT EXECUTE
                *               EXERCISE SCAN CONVERTER (15MZ)
                *               WORD 1 AND WORD 46 SET UP BY SWCHNRT WITH FBIT = 1
                *               GENERATE PRF SYNC. FOR RADAR MODES
                *
        17640   BITXEQ  EQU     $
17640 64110             LD      A0,FBIT
17641  5001             BNZ     $+2
17642 174120            RTS                     NOT BIT MODE
                *
                *               BIT MODE SELECTED
                *
        17643   BITX3   EQU     $
17643 64365             LD      A0,BITFREEZ
17644  5001             BNZ     $+2
17645 174120            RTS                     FREEZE, SKIP UPDATE
                *       MOVE SENSOR FOR SENSOR POSITION INTERRUPTS
17646 34002             LI      A0,STEP
17647 64772             LD      A1,BITDIR
17650  4401             BZ      $+2             CHECK DIRECTION
17651 170140            RNEG    A0,A0           L-R
17652 65363             LD      A2,BITAZC
17653 171020            RADD    A2,A0           AZ=AZ+1
17654  71007            CPI     A2,AZR          RIGHT EDGE CHECK
17655  7004             BN      BITX4           TURN TO R-L
17656  71067            CPI     A2,AZL          LEFT EDGE CHECK
17657  7006             BN      BITX5           OK, CONTINUE
17660 34002             LI      A0,STEP         TURN TO L-R
17661  5001             JMP     $+2
17662 34000     BITX4   LI      A0,0            R-L
17663 60372             ST      A0,BITDIR       NEW DIRECTION
17664 61363             ST      A2,BITAZC
17665  5755             JMP     BITX3           TRY NEW DIRECTION
17666 61363     BITX5   ST      A2,BITAZC       NEW AZ COUNTER VALUE
17667 171021            RADD    A2,A1           DIRECTION SHIFT OFFSET
17670 175220            SHL     A2,1            DOUBLE VIDEO FREQ.
17671 61373             ST      A2,BITVID
17672 65363             LD      A2,BITAZC
17673 175222            SHL     A2,3            SCALE TO FBIT
17674 34403             LI      A1,BITWD46
17675 171101            ROR     A2,A1
17676 61110             ST      A2,FBIT
                *
                *                               SET WORD 1 (VIDEO)
        17677   WORD1   EQU     $
17677 34400             LI      A1,0            PRESET AGC AND CLAMP OFF
17700 174627            SHL     A1,6
17701 134026            LDR     A0,XC0          BIT MODE, BIT VIDEO SELECT
17702  44373            OR      A0,BITVID       BIT VIDEO
17703 170101    WRD1B   ROR     A0,A1
17704 174225            SHL     A0,6
17705 170341    WRD1A   SOUT    A0,1            SEND WORD 1
17706  64111            LD      A0,SCMODE
17707  70001            CPI     A0,1            CHECK FOR SINGLE TARGET TRACK
17710  4412             BZ      BITX6           STT - SKIP SENSOR POS
                *
                *               MOVE SENSOR
                *
                *                               FBIT CONTAINS WORD 46 RIGHT SHIFTED 7
                *                               WHICH INCLUDES POSITION, BIT, PRF SEL,
                *                               BITDIR CONTAINS DIRECTION
                *
```

```
274   17711   63110   SENMOV   LD      A2,FBIT
275   17712   175226           SHL     A2,7
276   17713   171357           SOUT    A2,X'F'          SEND WORD #6
277   17714   134012   SENMOV1 LDR     A0,DELAY         DELAY FOR PHF READING
278   17715    74377           BINC    A0,S
279   17716   64372            LD      A0,BITDIR        SET SENSOR DIRECTION
280   17717     4402           HZ      S+3
281   17720   171354           SOUT    A2,X'C'
282   17721     5401           JNP     S+2
283   17722   171357           SOUT    A2,X'F'
284         17723    BITX6    EQU     S
285   17723   64366            LD      A0,BITCNT        YES
286   17724   30001            ADI     A0,1
287   17725   60366            ST      A0,BITCNT        BITCNT=BITCNT+1
288   17726   174120           RTS                      EXIT
289   17727   177406   DELAY   DATA    -250             560 MICRO,SEC, FOR BINC
290   17750      300   XE0     DATA    X'C0'
```

```
                                *
                    17731       MODETABL EQU     $           MODE TABLE .
17731   40050                   DATA     X'4028'    MODE  1  RDR
17732   62002                   DATA     X'6402'    MODE  6  RDY
17733   63002                   DATA     X'6602'    MODE  6  RDR  A
17734   63200                   DATA     X'6680'    MODE  6  RDR  X
17735   62200                   DATA     X'6480'    MODE  6  RDR  B
17736     154                   DATA     X'006C'    MODE  2  IR
            6           LSTFRM  EQU      $-MODETABL
      376                       SLOC     X'FE'
  376   17400                   DATA     BITSET
  377   17640                   DATA     BITXEQ
                                END
                                                    **END OF PASS TWO**

NO ERRORS

    211 (   323 OCT) WORDS OF CODE GENERATED
```

```
                **MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION**

                                **END OF PASS ONE**

NO ERRORS

**SYMBOL VALUES:

  4 S1      /      60  1        BASE    /      40  1        VAL     /  1315  1        VALADR  /  1316  1
  4 SBR     /    1317  1        SBIR    /     100  1        SBRR    /  1007  1        NEXT0   /  1050  1
  4 NEXT1   /    1055  1        NEXT2   /    1063  1        NEXT3   /  1070  1        NEXT4   /  1076  1
  4 NEXT5   /    1103  1        NEXT6   /    1113  1        NEXT7   /  1122  1        NEXT8   /  1131  1
  4 NEXT9   /    1136  1        LIM1    /    1010  1        SKPB    /  1006  1        SKIPA   /  1160  1
  4 SKIPB   /    1162  1        LOOP1   /    1042  1        EQ1     /  1235  1        CURNT   /  1000  1
  4 NXT1    /    1002  1        NXT3    /    1003  1        NXT5    /  1024  1        NXT7    /  1005  1
FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED
```

```
                        *          PROG MPROG INSTRUCTION SET
                        *VARIABLE ASSIGNMENTS
                        *BASE PAGE
                                DATA    X'7F'
                                DATA    X'AB'
                                SLOC    X'20'
                        BASE    DATA    X'1'
                                DATA    X'FF'
                                DATA    X'FFF'
                                DATA    X'FFFF'         -1
                                DATA    X'FFF1'         -F
                                DATA    X'FF01'         -FF
                                DATA    X'F001'         -FFF
                                DATA    X'F0F0'
                                DATA    X'5000'
                                SLOC    X'30'
                        S1      RES     16
                                SLOC    X'40'
                        SBIR    DATA    SBR
                                SLOC    X'200'
                        CURNT   DATA    X'0F0F'
                                DATA    X'F0F0'
                        NXT1    DATA    NEXT1
                        NXT3    DATA    NEXT3
                        NXT5    DATA    NEXT5
                        NXT7    DATA    NEXT7
                        SKPB    DATA    SKIPB
                        SBRR    DATA    SBR
                        LIM1    DATA    -5
                                SLOC    X'210'
                        *LOAD IMMEDIATE TEST
                                LI      A0,X'7F'
                        *ADD TEST
                                LI      A0,0
                                LI      A1,0
                                LI      A2,0
                                LI      X1,X'20'
                                LI      X2,X'10'
                                ADD     A0,0            ADD DIRECT
                                ADD     A1,0,X1         ADD INDEXED X1
                                ADD     A1,+4,X1
                                ADD     A2,X'11',X2     ADD INDEXED X2
                                ADD     A2,X'15',X2
                        *ADD IMMEDIATE TEST
                                LI      A0,0
                                ADI     A0,X'7F'
                        *AND TEST
                                LI      A0,X'70'
                                LI      A1,X'1F'
                                AND     A0,0            AND DIRECT
                                AND     A1,+4,X1        AND INDEXED
                        *INCREMENT AND BRANCH IF NOT ZERO TEST
                                LI      A0,-5
                        LOOP1   RINC    A0,LOOP1
                        *BRANCH NEGATIVE TEST
                                LI      A0,0
                                BN      NEXT0           BRANCH NEGATIVE RELATIVE
                                LI      A0,-128
                                BN      NEXT0
                                ADI     A0,1
```

```
 62   1050   34000  NEXT0   LI      A0,0
 63   1051    5330          BN      *NXT1          BRANCH NEGATIVE RELATIVE INDIRECT
 64   1052   34200          LI      A0,-128
 65   1053    5326          BN      *NXT1
 66   1054   30001          ADI     A0,1
 67   1055    5400  NEXT1   NOP
 68                *BRANCH NOT ZERO TEST
 69   1056   34000          LI      A0,0
 70   1057    5003          BNZ     NEXT2          BRANCH NOT ZERO RELATIVE
 71   1060   34001          LI      A0,1
 72   1061    5001          BNZ     NEXT2
 73   1062   30377          ADI     A0,-1
 74   1063   34000  NEXT2   LI      A0,0
 75   1064    1316          BNZ     *NXT3          BRANCH NOT ZERO RELATIVE INDIRECT
 76   1065   34001          LI      A0,1
 77   1066    1314          BNZ     *NXT3
 78   1067   30377          ADI     A0,-1
 79   1070    5400  NEXT3   NOP
 80                *BRANCH POSITIVE TEST
 81   1071   34200          LI      A0,-128
 82   1072    6403          BP      NEXT4          BRANCH POSITIVE RELATIVE
 83   1073   34001          LI      A0,1
 84   1074    6401          BP      NEXT4
 85   1075   30377          ADI     A0,-1
 86   1076   34200  NEXT4   LI      A0,-128
 87   1077    2704          BP      *NXT5          BRANCH POSITIVE RELATIVE INDIRECT
 88   1100   34001          LI      A0,1
 89   1101    2702          BP      *NXT5
 90   1102   30377          ADI     A0,-1
 91   1103    5400  NEXT5   NOP
 92                *BRANCH ZERO TEST
 93   1104   34377          LI      A0,-1
 94   1105    4405          BZ      NEXT6          BRANCH ZERO RELATIVE
 95   1106   34001          LI      A0,1
 96   1107    4403          BZ      NEXT6
 97   1110   34000          LI      A0,0
 98   1111    4401          BZ      NEXT6
 99   1112   30001          ADI     A0,1
100   1113   34377  NEXT6   LI      A0,-1
101   1114     670          BZ      *NXT7          BRANCH ZERO RELATIVE INDIRECT
102   1115   34001          LI      A0,1
103   1116     665          BZ      *NXT7
104   1117   34000          LI      A0,0
105   1120     664          BZ      *NXT7
106   1121   30001          ADI     A0,1
107   1122    5400  NEXT7   NOP
108                *COMPARE TEST
109   1123   34017          LI      A0,X'F'
110   1124   54000          CMPR    A0,0           COMPARE DIRECT
111   1125    4403          BZ      NEXT8
112   1126  114001          CMPR    A0,+1,X1       COMPARE INDEXED
113   1127    4401          BZ      NEXT8
114   1130   30001          ADI     A0,1
115   1131    5400  NEXT8   NOP
116                *COMPARE IMMEDIATE TEST
117   1132   34005          LI      A0,5
118   1133   70005          CPI     A0,5
119   1134    4401          BZ      NEXT9
120   1135   30373          ADI     A0,-5
121   1136    5400  NEXT9   NOP
122                *DOUBLE PRECISION ADD TEST
123   1137   34177          LI      A0,X'7F'
124   1140   34000          LI      A1,0
125   1141   50000          DADD    A0,0           DOUBLE PRECISION ADD DIRECT
126   1142  110005          DADD    A0,+5,X1       DOUBLE PRECISION ADD INDEXED
```

```
127                     *DIVIDE TEST
128     1143    34037           LI      A0,X'1F'
129     1144    34400           LI      A1,0
130     1145    56000           DIV     A0,0            DIVIDE DIRECT
131     1146    34177           LI      A0,X'7F'
132     1147    34400           LI      A1,0
133     1150    116002          DIV     A0,+2,X1        DIVIDE INDEXED
134                     *DOUBLE PRECISION SHIFT LEFT TEST
135     1151    34017           LI      A0,X'F'
136     1152    34417           LI      A1,X'F'
137     1153    174202          DSHL    A0,3
138                     *DOUBLE PRECISION SHIFT RIGHT
139     1154    174262          DSHR    A0,3
140                     *JUMP TEST
141     1155    5402            JMP     SKIPA           JUMP RELATIVE
142     1156    5400            NOP
143     1157    5400            NOP
144     1160    1625    SKIPA   JMP     *SKPB           JUMP RELATIVE INDIRECT
145     1161    30201           ADI     A0,1
146     1162    5400    SKIPB   NOP
147                     *JUMP SUB-ROUTINE TEST
148     1163    6133            JSR     SBR
149     1164    2222            JSR     *SHRR
150     1165     100            JSRZ    *X'40'
151                     *LIMIT TEST
152     1166    34010           LI      A0,8
153     1167    130220          LIM     A2,LIM1
154     1170    34003           LI      A0,3
155     1171    130216          LIM     A0,LIM1
156     1172    54376           LI      A0,-2
157     1173    130214          LIM     A0,LIM1
158     1174    34370           LI      A2,-8
159     1175    130212          LIM     A0,LIM1
160                     *MULTIPLY TEST
161     1176    34003           LI      A0,3
162     1177    52000           MULT    A0,0            MULTIPLY DIRECT
163     1200    112001          MULT    A0,1,X1         MULTIPLY-INDEXED
164                     *LOGICAL OR TEST
165     1201    34200           LI      A0,-128
166     1202    44000           OR      A0,0            OR DIRECT
167     1203    34200           LI      A0,-128
168     1204    104000          OR      A0,0,X1         OR INDEXED
169                     *PARALLEL INPUT TEST
170     1205    171701          PIN     A3,1
171                     *PUSH STACK TEST
172     1206    127011          LD      SP,+9,X1
173     1207    34037           LI      A0,X'1F'
174     1210    170166          PUSH    SP,A0
175                     *POP STACK TEST
176     1211    34000           LI      A0,0
177     1212    170206          POP     SP,A0
178                     *PARALLEL OUTPUT TEST
179     1213    171721          POUT    A3,1
180                     *ABSOLUTE VALUE TEST
181     1214    34000           LI      A0,0
182     1215    54773           LI      A1,-5
183     1216    174141          RABS    A0,A1
184                     *REGISTER ADD TEST
185     1217    34005           LI      A0,5
186     1220    54412           LI      A1,10
187     1221    174021          RADD    A2,A1
188                     *REGISTER AND TEST
189     1222    34035           LI      A0,X'10'
190     1223    54536           LI      A1,X'5E'
191     1224    174041          RAND    A0,A1
```

```
192                         *REGISTER COMPARE TEST
193   1225   35573            LI      A0,-5
194   1226   34405            LI      A1,5
195   1227   35573            LI      A2,-5
196   1230   170121           RCMP    A0,A1
197   1231     4403           BZ      EQ1
198   1232   170122           RCMP    A0,A2
199   1233     4401           BZ      EQ1
200   1234   30005            ADI     A0,5
201   1235     5400   EQ1      NOP
202                         *REGISTER MOVE TEST
203   1236   34005            LI      A0,5
204   1237   34412            LI      A1,10
205   1240   170001           RMOV    A0,A1
206                         *REGISTER NEGATE TEST
207   1241   34005            LI      A0,5
208   1242   34412            LI      A1,10
209   1243   170141           RNEG    A0,A1
210                         *REGISTER ONE'S COMPLEMENT TEST
211   1244   34000            LI      A0,0
212   1245   34600            LI      A1,-128
213   1246   174101           RNOT    A0,A1
214                         *REGISTER OR TEST
215   1247   34074            LI      A0,X'3C'
216   1250   34570            LI      A1,X'78'
217   1251   170101           ROR     A0,A1
218                         *ROTATE TEST
219   1252   34017            LI      A0,X'F'
220   1253   174243           ROT     A0,4
221   1254   174243           ROT     A0,4
222   1255   174247           ROT     A0,8
223                         *REGISTER SUBTRACT TEST
224   1256   34005            LI      A0,5
225   1257   34412            LI      A1,10
226   1260   170041           RSUB    A0,A1
227                         *SHIFT LEFT TEST
228   1261   34017            LI      A0,X'F'
229   1262   174223           SHL     A0,4
230                         *SHIFT RIGHT TEST
231   1263   170223           SHR     A0,4
232                         *SHIFT RIGHT ARITHMETIC TEST
233   1264   34063            LI      A0,X'33'
234   1265   170243           SHRA    A0,4
235   1266   124407           LD      A1,7,X1
236   1267   170645           SHRA    A1,6
237                         *SKIP IF AND EQUAL ZERO TEST
238   1270   34160            LI      A0,X'70'
239   1271   12000            SKAZ    A0,0            SKIP DIRECT
240   1272   14001            SKAZ    A0,+1,X1        SKIP INDEXED
241   1273   30017            ADI     A0,X'F'
242                         *SKIP IF I/O READY
243   1274   174365           SKR     5
244   1275   30005            ADI     A0,5
245   1276     5400           NOP
246                         *LOAD TEST
247   1277   64000            LD      A0,0            LOAD DIRECT
248   1300   124000           LD      A0,0,X1         LOAD INDEXED
249   1301   134013           LDR     A0,VAL          LOAD RELATIVE
250   1302    24413           LDR     A1,*VALADR      LOAD RELATIVE INDIRECT
251                         *STORE TEST
252   1303   34017            LI      A0,X'F'
253   1304   60003            ST      A0,3            STORE DIRECT
254   1305   64003            LD      A1,3
255   1306   120020           ST      A0,X'10',X1     STORE INDEXED
256   1307   125060           LD      A2,X'30',X1
257   1310   20005            ST      A0,*VALADR      STORE RELATIVE
258   1311   25404            LDR     A3,*VALADR
```

```
                        *SUBTRACT TEST
1312    34177           LI      A0,X'7F'
1313    40000           SUB     A0,0            SUBTRACT DIRECT
1314    100002          SUB     A0,*+2,X1       SUBTRACT INDEXED
                        *DATA AREA
1315    170017  VAL     DATA    X'F00F'
1316     1315   VALADR  DATA    VAL
                        *SUBROUTINE PAGE
1317    174120  SBR     RTS
                        *STACK AREA
1320    170360          DATA    X'F0F0'
 1465                   RES     100
                        $END
                         END
                                        **END OF PASS TWO**

NO ERRORS

     331 (   513 OCT) WORDS OF CODE GENERATED
```

END OF PASS ONE

NO ERRORS

**SYMBOL VALUES:

4 BADCHAR	77070	1	LAST	5	1	MASK	1	1	RAMPT0	76144	1
4 RAMPT1	76322	1	RAMPT2	77202	1	RAMPT3	77622	1	RAMP1A	76466	1
4 SAV2ND	76430	1	SAMELINE	76310	1	SAVERST	76424	1	VARADDR	1640	1
4 ABORT	77451	1	OBGSTK	76143	1	KBWAIT	77460	1	S8A	42	1
4 SCRATCH	1640	1	SCRCH3	36	1	SCRCH1	34	1	SCRCH2	35	1
4 LOLIMREL	76567	1	LOSTPIND	76553	1	LOSTAT	76655	1	MDS1	77511	1
4 MDSI1	76332	1	MDSO	77512	1	DEFAULT	77430	1	DELETALL	77157	1
4 DEBUGSTK	66	1	GETC	77365	1	GETC0	76145	1	GETC1	76323	1
4 GETC2	77203	1	GETHX1	76330	1	GETHX2	77210	1	GETHEX	77303	1
4 HEXALPH	77252	1	HEXNUM	77260	1	REG0	10	1	REG1	11	1
4 REG2	12	1	REG3	13	1	REG4	14	1	REG5	15	1
4 REG6	16	1	REG7	17	1	RESEF	77172	1	RESET3	77624	1
4 REGLOOP	76177	1	REMOVTRP	77150	1	RESREG	76700	1	RETRN1	76657	1
4 RETRN2	75660	1	RESTADDR	76467	1	RESTOR	76215	1	SETINSTR	76560	1
4 SECOND	4	1	SETRPAD	76766	1	SETRAP	76522	1	SETRTRNS	76641	1
4 SETSTACK	76674	1	LFEED	76140	1	LFTERM	76357	1	MFDR	77515	1
4 XFF	77625	1	XFFF	77250	1	CHKARROW	77326	1	CHKLF	77317	1
4 CHKTRAP	77164	1	SHOWALL	76174	1	BINCADDR	76664	1	BINCINST	76742	1
4 BINHX1	76327	1	BINHX2	77207	1	BINHEX	77266	1	DISPLAY	76305	1
4 DISPREG	76152	1	DISPTRAP	77014	1	DISPMSK	76663	1	FIRST	3	1
4 JINSTRC	76765	1	SIMULAT	76652	1	TINSTPTR	4	1	SKRAZLOK	77003	1
4 SKZLCKAD	76666	1	SKRATS	76773	1	SKSTSAD	76663	1	ALLTRAPS	77043	1
4 CLRPWREL	177	1	CLRTRAP	177	1	ILLEG	77345	1	CMNDTBL	76201	1
4 IMSKINST	76645	1	JMPBRNCH	76575	1	JMPINST	76716	1	CNVRT	77365	1
4 INTERPRT	76120	1	INCH	77431	1	INPPPS	77623	1	INSRTMC	77531	1
4 INTRP1	76551	1	INTRP2	77211	1	INIT	76065	1	INSTMSK	76661	1
4 PNCHDATA	46	1	PNCHBC	77472	1	PNCHKCNT	43	1	PNCHADDR	44	1
4 PNCHMEM	77516	1	PNCHBK	77422	1	PNCHBUK	77474	1	PNCHKSUM	45	1
4 PNCHSTRT	77605	1	CONTINU	76266	1	CONVERT	7	1	COPYMEM	76334	1
4 FOUNDTRP	77136	1	FORK	77251	1	GOODHEX	77358	1	GOTOINIT	76230	1
4 LOC	6	1	MONE	77513	1	MORMEM	76364	1	MOVNXT	76337	1
4 MOIOC	76333	1	NOPREV	77242	1	NOSUCCES	76676	1	TOOMUCH	77576	1
4 WORD	2	1	OPENRD	76235	1	OPENMEM	76346	1	OPENLOC	76413	1
4 OPENPREV	76406	1	OPENMSK	76224	1	OPENXT	76401	1	OPENCUR	76376	1
4 SPACER	77567	1	TRADR1	76653	1	TRADR2	76654	1	TPINT	4	1
4 TPINTADR	76150	1	ARQWTERM	76371	1	BRNCHMSK	76662	1	CRLF	77560	1
4 CRLFR	76147	1	CRLF1	76326	1	CRLF2	77226	1	CRLF3	77626	1
4 CRET	76141	1	ORIGIN	76250	1	PREVOP	77231	1	SRCHMEM	76253	1
4 TRPINSTR	76665	1	TRAP	170	1	TRAPADDR	20	1	TRAPINST	24	1
4 TRAPLOOP	77046	1	TRAPRTRN	77073	1	JSRINST	76727	1	JSTRUC	76656	1
4 JSRZINST	76707	1	RSTREG	76664	1	RSTRINST	77143	1	RSTRTRAP	76652	1
4 USRINSTR	31	1	USALEVEL	41	1	USRGPRND	33	1	USRPC	40	1
4 USPSTAT	37	1	CTABL	76142	1	MTMD	77514	1	RTNCNTRL	76674	1
4 STATINST	32	1	STATRTRN	76530	1	TTY	36	1	TTYIN	176	1
4 TTYOUT	175	1	CONTRAP	50	1	NULTRAP	76151	1	NULTRPAD	77201	1
4 PUTC	77407	1	PUTC0	76146	1	PUTC1	76324	1	PUTC2	77204	1
4 QUEST2	77205	1	QUESTION	77354	1	QUESTN1	76325	1	SUCCESS	76667	1
4 SUBURTRN	76767	1	SUMPREV	77245	1	PWRFL	170	1	EXECUTE	76434	1
4 EXECNXT	76470	1	HXADD	77212	1	HXRESULT	77222	1	HXARITH	0	1
4 HXSUB	77216	1	NXTBLNK	77435	1	NXTBLOC	77522	1	NXTDIGIT	77270	1
4 NXTHEX	77307	1	NXTMEM	76347	1	NXTPNCH	77475	1	NXTRP	76454	1
3 NXTRAP	77122	1	NXTSRCH	76262	1	NXTWORD	77551	1			

FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED

```
                        *
        76000   ORIGIN  EQU     X'7C00'         PROGRAM ORIGIN
                        **********************************
                        **********************************
                        ******                      ******
                        ******  MMP DEBUG PACKAGE    ******
                        ******  OREN YUEN  3/8/76    ******
                        ******                      ******
                        ******  HUGES AIRCRAFT CO    ******
                        ******                      ******
                        **********************************
                        **********************************
                        *
                                SLOC    4
    4   77073   TPINT   DATA    TRAPRTN         ADDRESS OF TRAP INTERRUPT
                        *
                        *       INTERFACE ADDRESS ASSIGNMENTS
                        *
          170   PWRFL   EQU     X'78'
          177   CLRPWRFL EQU    X'7F'
           36   TTY     EQU     X'1E'
          176   TTYIN   EQU     X'7E'
          175   TTYOUT  EQU     X'7D'
          170   TRAP    EQU     PWRFL
          177   CLRTRAP EQU     CLRPWRFL
         1640   SCRATCH EQU     X'340'          SCRATCH PAD AND STACK AREA
    1640        SLOC    SCRATCH
```

```
                        *
                        *       VARIABLES
                        *
1640            VARADDR EQU     $               ADDRESS OF RAM AREA
   0            HXARITH EQU     $-VARADDR
1641                    RES     1               HEX ARITH FLAG
   1            MASK    EQU     $-VARADDR
1642                    RES     1               SEARCH MASK
   2            WORD    EQU     $-VARADDR
1643                    RES     1               SEARCH WORD
   3            FIRST   EQU     $-VARADDR
1644                    RES     1               ADDR OF 1ST MEM LOC TO OPEN/PUNCH/COPY
   4            SECOND  EQU     $-VARADDR
1645                    RES     1               ADDR OF LAST MEM LOC TO BE COPIED
   5            LAST    EQU     $-VARADDR
1646                    RES     1               ADDRESS OF LAST WORD TO BE PUNCHED
   6            LOC     EQU     $-VARADDR
1647                    RES     1               LOCATION REGISTER
   7            CONVERT EQU     $-VARADDR
1650                    RES     1               CONVERSION FLAG (0,..9,A,..F RCVD)
  10            REG0    EQU     $-VARADDR       *
1651                    RES     1               *
  11            REG1    EQU     $-VARADDR       *
1652                    RES     1               *
  12            REG2    EQU     $-VARADDR       *
1653                    RES     1               * USER'S
  13            REG3    EQU     $-VARADDR       * REGISTERS
1654                    RES     1               *
  14            REG4    EQU     $-VARADDR       *
1655                    RES     1               *
  15            REG5    EQU     $-VARADDR       *
1656                    RES     1               *
  16            REG6    EQU     $-VARADDR       *
1657                    RES     1               *
  17            REG7    EQU     $-VARADDR       *
1660                    RES     1               **
  20            TRAPADDR EQU    $-VARADDR
1664                    RES     4               TRAP ADDRESSES
  24            TRAPINST EQU    $-VARADDR
1670                    RES     4               TRAP INSTRUCTIONS
  30            CURTRAP EQU     $-VARADDR
1671                    RES     1               CURRENT TRAP NUMBER
```

```
 72                31      USRINSTR EQU     $-VARADDR      **
 73          1672                 RES     1              *
 74                32      STATINST EQU     $-VARADDR      *
 75          1673                 RES     1              *
 76                33      USROPAND EQU     $-VARADDR      *
 77          1674                 RES     1              *
 78                34      SCRCH1   EQU     $-VARADDR      *
 79          1675                 RES     1              * WORK AREA
 80                35      SCRCH2   EQU     $-VARADDR      *
 81          1676                 RES     1              * FOR EXECNAT
 82                36      SCRCH3   EQU     $-VARADDR      *
 83          1677                 RES     1              * COMMAND
 84                37      USRSTAT  EQU     $-VARADDR      *
 85          1700                 RES     1              *
 86                40      USRPC    EQU     $-VARADDR      *
 87          1701                 RES     1              *
 88                41      USRLEVEL EQU     $-VARADDR      *
 89          1702                 RES     1              **
 90                42      SBA      EQU     $-VARADDR
 91          1703                 RES     1              SUCCESSFUL BRANCH ADDRESS
 92                43      PNCHWCNT EQU     $-VARADDR       **
 93          1704                 RES     1              - WORD COUNT *
 94                44      PNCHADDR EQU     $-VARADDR       *
 95          1705                 RES     1              ADDRESS * PUNCH
 96                45      PNCHKSUM EQU     $-VARADDR      * BUFFER
 97          1706                 RES     1              CHECKSUM *
 98                46      PNCHDATA EQU     $-VARADDR      *
 99          1725                 RES     16             DATA **
100                66      DEBUGSTK EQU     $-VARADDR
101          1746                 RES     14             DEBUG STACK AREA
102         76000                 SLOC    ORIGIN
103         76000     5464  GOTOINIT JMP     INIT
```

```
107                     *
108                     ************************************
109                     *        DEBUG COMMAND TABLE       *
110                     ************************************
111                     *
112   76001   77212  CMNDTBL  DATA    HXADD       +
113   76002   76424          DATA    SAVFRST     /
114   76003   77216          DATA    HXSUB       -
115   76004   76376          DATA    OPENCUR     .
116   76005   76346          DATA    OPENMEM     /
117   76006   77260          DATA    HEXNUM      0 **
118   76007   77260          DATA    HEXNUM      1 *
119   76010   77260          DATA    HEXNUM      2 *
120   76011   77260          DATA    HEXNUM      3 *
121   76012   77260          DATA    HEXNUM      4 * HEX
122   76013   77260          DATA    HEXNUM      5 * CHARS
123   76014   77260          DATA    HEXNUM      6 *
124   76015   77260          DATA    HEXNUM      7 *
125   76016   77260          DATA    HEXNUM      8 *
126   76017   77260          DATA    HEXNUM      9 **
127   76020   77354          DATA    QUESTION    :
128   76021   76334          DATA    COPYMEM     SEMICOLON
129   76022   77354          DATA    QUESTION    <
130   76023   77222          DATA    HXRESULT    =
131   76024   76430          DATA    SAV2ND      >
132   76025   77354          DATA    QUESTION    QUESTIONMARK
133   76026   77354          DATA    QUESTION    @
134   76027   77252          DATA    HEXALPH     A **
135   76030   77252          DATA    HEXALPH     B *
136   76031   77252          DATA    HEXALPH     C * HEX
137   76032   77252          DATA    HEXALPH     D * CHARS
138   76033   77252          DATA    HEXALPH     E *
139   76034   77252          DATA    HEXALPH     F **
140   76035   77354          DATA    QUESTION    G
141   76036   77422          DATA    PNCHBK      H
142   76037   77354          DATA    QUESTION    I
143   76040   77354          DATA    QUESTION    J
144   76041   77354          DATA    QUESTION    K
145   76042   76413          DATA    OPENLOC     L
146   76043   76224          DATA    OPENMSK     M
147   76044   76470          DATA    EXECNXT     N (CONTINUE FROM TRAP)
148   76045   77354          DATA    QUESTION    O
149   76046   77516          DATA    PNCHMEM     P
150   76047   77354          DATA    QUESTION    Q
151   76050   76152          DATA    DISPREG     R
152   76051   76253          DATA    SRCHMEM     S
153   76052   77014          DATA    DISPTRAP    T (BRKPNT)
154   76053   77150          DATA    REMOVTRP    U
155   76054   77354          DATA    QUESTION    V
156   76055   76235          DATA    OPENXO      W
157   76056   76434          DATA    EXECUTE     X
158   76057   77354          DATA    QUESTION    Y
159   76060   77603          DATA    PNCHSTRT    Z
160   76061   77354          DATA    QUESTION    LEFT SQU BRACKET
161   76062   77354          DATA    QUESTION    SLASH
162   76063   77354          DATA    QUESTION    RIGHT SQU BRACKET
163   76064   76406          DATA    OPENPREV    UPARROW
```

```
165                     *
166                     *       INITIALIZATION
167                     *
168   76065  173777  INIT    IOCP    CLRTRAP         CLEAR TRAP INTERRUPT
169   76066    3403           IMSK    3               MASK OFF LEVELS 0-3
170   76067  137053           LDR     SP,DBGSTK
171   76070  136453           LDR     X2,RAMPT0
172   76071   35000           LI      2,0
173   76072   35400           LI      3,0
174   76073  161027           ST      2,CONVERT,X2
175   76074  161000           ST      2,HXARITH,X2
176   76075  161001           ST      2,MASK,X2
177   76076  161002           ST      2,WORD,X2
178   76077  161003           ST      2,FIRST,X2
179   76100  163016           ST      3,REG6,X2
180   76101  134446           LDR     1,TPINTADR
181   76102   62404           ST      1,TPINT
182   76103   34104           LI      0,X'44'
183   76104    2041           JSR     *PUTC0          OUTPUT D
184   76105   34105           LI      0,X'45'
185   76106    2037           JSR     *PUTC0          OUTPUT E
186   76107   34102           LI      0,X'42'
187   76110    2035           JSR     *PUTC0          OUTPUT B
188   76111   34125           LI      0,X'55'
189   76112    2033           JSR     *PUTC0          OUTPUT U
190   76113   34107           LI      0,X'47'
191   76114    2031           JSR     *PUTC0          OUTPUT G
192   76115   34007           LI      0,7
193   76116    2027           JSR     *PUTC0          OUTPUT DING
194   76117    2027           JSR     *CRLF0
```

```
196                         *
197                         *       INTERPRETER DECODES CHARACTERS RECEIVED FROM THE KEYBRD
198                         *       AND JUMPS TO THE APPROPRIATE ROUTINE TO HANDLE IT.
199                         *       CHARACTER RECEIVED IS KEPT IN REGISTER 0.
200                         *
201   76120   136423  INTERPRT LDR     X2,RAMPT0
202   76121     2023           JSR     *GETC0          GET CHARACTER
203   76122    70012           CPI     0,X'0A'         CHECK FOR LF
204   76123      414           BZ      *LFEED
205   76124    70015           CPI     0,X'0D'         CHECK FOR CR
206   76125      413           BZ      *CRET
207   76126    70137           CPI     0,X'5F'         CHECK FOR GREATER THAN UPPER BOUND
208   76127     2575           BP      *QUESTN1
209   76130    54453           LI      1,X'2B'
210   76131   170041           RSUB    0,1             CHECK FOR LESS-THAN-LOWER BOUND-(+)
211   76132     5172           BR      *QUESTN1
212   76133   172000           RMOV    X1,0
213   76134   134405           LDR     1,CTABL
214   76135   172021           RADD    X1,1
215   76136   127400           LD      PC,0,X1
216   76137     5400           NOP
217   76140    76401  LFEED    DATA    OPENXT
218   76141    77172  CRET     DATA    RESET
219   76142    76201  CTABL    DATA    CMNDTBL
220   76143     1726  DEBGSTK  DATA    DEBUGSTK+VARADDR
221   76144     1540  RAMPT0   DATA    VARADDR
222   76145    77365  GETC0    DATA    GETC
223   76146    77427  PUTC0    DATA    PUTC
224   76147    77360  CRLF0    DATA    CRLF
225   76150    77073  TPINTADR DATA    TRAPRTRN
226   76151    76151  NULTRAP  DATA    $
```

```
                        *
                        *       DISPREG CHECKS CONVERT FLAG TO DETERMINE WHETHER A SINGLE REG
                        *       (POINTED TO BY R2) IS TO BE OPENED OR ALL REGISTERS ARE TO BE
                        *       DISPLAYED.  THEN CARRIES OUT THE REQUIRED OPERATION.
                        *
 76152  164007 DISPREG  LD      0,CONVERT,X2
 76153    4420          BZ      SHOWALL         IF NO HEX DIGIT ENTERED, DISP ALL REGS
 76154   71000          CPI     2,0             MAKE SURE R2 CONTAINS VALID REG#
 76155    6401          BP      $+2
 76156    1546          JMP     *QUESTN1        LESS THAN 0
 76157   71010          CPI     2,8
 76160    7001          BN      $+2
 76161    1543          JMP     *QUESTN1        GREATER THAN 7

 76162  34040           LI      0,X'20'
 76163    2140          JSR     *PUTC1          OUTPUT *SPACE
 76164  172422          RADD    X2,2
 76165   32010          ADI     X2,REG0         COMPUTE PNTR TO REG IN RAM AREA
 76166  165400          LD      3,0,X2
 76167    2137          JSR     *BINHX1         OUTPUT REGISTER PNTED TO BY R2
 76170    2137          JSR     *GETHX1
 76171   70401          CPI     1,1             CHECK FOR CR TERMINATOR
 76172    1132          BNZ     *QUESTN1
 76173    5421          JMP     RESTOR

 76174    2131 SHOWALL  JSR     *CRLF1          OUTPUT CR & LF
 76175   35370          LI      2,-8            R2 WILL = LOOP CNTR
 76176   32410          ADI     X2,REG0         COMPUTE RAM PTR TO REG

 76177   34122 REGLOOP  LI      0,X'52'
 76200    2123          JSR     *PUTC1          OUTPUT R
 76201   34070          LI      0,X'38'         COMPUTE
 76202  170022          RADD    0,2             REG#
 76203    2122          JSR     *PUTC1          OUTPUT REG#
 76204   34040          LI      0,X'20'
 76205    2116          JSR     *PUTC1          OUTPUT *SPACE
 76206  165400          LD      3,0,X2
 76207    2117          JSR     *BINHX1         OUTPUT REG CONTENTS
 76210   34040          LI      0,X'20'
 76211    2112          JSR     *PUTC1          OUTPUT
 76212    2111          JSR     *PUTC1          2 *SPACES
 76213   32021          ADI     X2,1
 76214   75362          BINC    2,REGLOOP
 76215    2110 RESTOR   JSR     *CRLF1          OUTPUT CR & LF
 76216  136503          LDR     X2,RAMPT1       RESTORE REGISTERS
 76217   35020          LI      2,0
 76220  161027          ST      2,CONVERT,X2
 76221  161020          ST      2,HXARITH,X2
 76222   35400          LI      3,0
 76223    5674          JMP     INTERPRT
```

```
279                     *
280                     *       OPENMSK DISPLAYS MASK CONTENTS AND WAITS FOR HEX INPUT
281                     *
282   76224   34040     OPENMSK LI      0,X'20'
283   76225   2076            JSR     *PUTC1          OUTPUT *SPACE
284   76226   165401          LD      3,MASK,X2
285   76227   2077            JSR     *BINHX1         OUTPUT MASK CONTENTS

287   76230   32401           ADI     X2,MASK         SET POINTER TO MASK
288   76231   2076            JSR     *GETHX1         GET & STORE HEX CHARACTERS

290   76232   70401           CPI     1,1             CHECK FOR CR TERMINATOR
291   76233   1071            BNZ     *QUESTN1
292   76234   5760            JMP     RESTOR


294                     *
295                     *       OPENWD DISPLAYS WORD AND WAITS FOR HEX INPUT
296                     *
297   76235   34040     OPENWD  LI      0,X'20'
298   76236   2065            JSR     *PUTC1          OUTPUT *SPACE
299   76237   165402          LD      3,WORD,X2
300   76240   2066            JSR     *BINHX1         OUTPUT WORD CONTENTS

302   76241   32402           ADI     X2,WORD         SET POINTER TO WORD
303   76242   2065            JSR     *GETHX1         GET & STORE HEX CHARACTERS

305   76243   70401           CPI     1,1             CHECK FOR CR TERMINATOR
306   76244   1060            BNZ     *QUESTN1
307   76245   34012           LI      0,X'0A'
308   76246   2055            JSR     *PUTC1          OUTPUT LF
309   76247   136452          LDR     X2,RAMPT1       RESTORE REGISTERS
310   76250   35000           LI      2,0
311   76251   35400           LI      3,0
312   76252   5645            JMP     INTERRT
```

```
315                         *
316                         *       SRCHMEM SEARCHES MEMORY FROM THE LOC IN FIRST THRU THE LOC
317                         *       IN R2 PERFORMING AN AND OPERATION BETW 'MASK' AND THE CONTENTS
318                         *       OF EACH LOC.  IF RE RESULT IS = TO 'WORD' THE ADDRESS AND THE
319                         *       FULL CONTENT OF THE LOCATION IS PRINTED OUT.
320                         *
321   76253   164007  SRCHMEM LD      0,CONVERT,X2    CHECK THAT FLAG IS SET
322   76254    72001          CPI     0,1
323   76255     1047          BNZ     *QUESTN1        IT WASNT
324   76256     2047          JSR     *CRLF1          OUTPUT CR & LF
325   76257    34772          LI      1,-6            R1 = # MEM LOCS TO BE PRINTED PER LINE
326   76260    31001          ADI     2,1             INCREMENT LAST LOC
327   76261   166003          LD      X1,FIRST,X2     X1 = POINTER TO MEMORY
328   76262   134000  NXTSRCH LDR     00,X1           R0 = CONTENTS OF MEM LOC TO BE TESTED
1   501
VALUE OF OPERAND IS OUT OF RANGE--SET TO 0
329   76263   146001          AND     0,MASK,X2       MASK THE CONTENTS
330   76264   154002          CMPR    0,WORD,X2       CHECK IF = TO WORD
331   76265     4417          BZ      DISPLAY         IT IS
332   76266   172166  CONTINU PUSH    SP,X1
333   76267   136042          LDR     X1,MOSI1
334   76270   134042          LDR     0,MOIOO
335   76271    14137          SKAZ    0,X'5F',X1
336   76272     5401          JMP     $+2
337   76273     5404          JMP     $+5
338   76274    34000          LI      0,0
339   76275   120137          ST      0,X'5F',X1
340   76276   172226          POP     SP,X1
341   76277     5715          JMP     RESTOR
342   76300   172226          POP     SP,X1
343   76301    32001          ADI     X1,1            GET NEXT
344   76302   172122          RCMP    X1,2            CHECK IF REACHED END
345   76303     7356          BN      NXTSRCH         NO, GET NEXT ONE
346   76304     5710          JMP     RESTOR          FINISHED

348   76305    74402  DISPLAY BINC    1,SAMELINE      CHECK COLUMN COUNT
349   76306    34772          LI      1,-6
350   76307     2016          JSR     *CRLF1          OUTPUT CR & LF
351   76310   171404  SAMELINE RMOV   3,X1            OUTPUT THE ADDRESS
352   76311     2015          JSR     *BINHX1
353   76312    34057          LI      0,X'2F'
354   76313     2010          JSR     *PUTC1          OUTPUT A SLASH
355   76314   125400          LD      3,0,X1
356   76315     2011          JSR     *BINHX1         OUTPUT THE CONTENTS
357   76316    34040          LI      0,X'20'         OUTPUT
358   76317     2004          JSR     *PUTC1          TWO
359   76320     2003          JSR     *PUTC1          *SPACES
360   76321     5744          JMP     CONTINU

362   76322     1640  RAMPT1  DATA    VARADDR
363   76323    77565  GETC1   DATA    GETC            ***NOT REQUIRED***
364   76324    77407  PUTC1   DATA    PUTC
365   76325    77554  QUESTN1 DATA    QUESTION
366   76326    77360  CRLF1   DATA    CRLF
367   76327    77266  BINHX1  DATA    BINHEX
368   76330    77535  GETHX1  DATA    GETHEX
369   76331    74120  INTRP1  DATA    INTERPRT
370   76332   117400  MOSI1   DATA    X'9F00'
371   76333      400  MOIOO   DATA    X'0100'
```

```
                        *
                        *       COPYMEM COPIES THE CONTENTS OF MEM BOUNDED BY THE LOC IN FIRST
                        *       AND SECOND INTO CONSECUTIVE LOCATIONS IN MEMORY STARTING AT THE
                        *       LOC POINTED TO BY R2.
                        *
                        *
76334  166023  COPYMEM  LD      X1,FIRST,X2     X1 = 1ST ADDR OF SOURCE AREA
76335  164424           LD      1,SECOND,X2     X1 = LAST ADDR OF SOURCE AREA
76336  172402           RMOV    X2,2            X2 = 1ST ADDR OF DESTINATION AREA

76337  124000  MOVNXT   LD      0,0,X1
76340  160000           ST      0,0,X2
76341  172121           RCMP    X1,1
76342    4652           BZ      RESTOR
76343   32001           ADI     X1,1
76344   32401           ADI     X2,1
76345    5771           JMP     MOVNXT



                        *
                        *       OPENMEM DISPLAYS MEM LOC PNTED TO BY R2 AND WAITS FOR HEX INPUT
                        *       TERMINATION OF OPERATION ON CURRENT LOCATION IS ACHIEVED
                        *       WITH A CR, LF, OR UP ARROW.
                        *
76346  172402  OPENMEM  RMOV    X2,2            X2 = ADDR OF MEM LOC BEING EXAMINED
76347  165400  NXTMEM   LD      3,0,X2          LOAD MEMORY CONTENTS
76350    2356           JSR     *BINHX1         DISPLAY IT
76351    6356           JSR     GETHX1
76352   70401           CPI     1,1             CHECK FOR CR
76353    5003           BNZ     LFTERM
76354  136345           LDR     X1,RAMPT1       CR,SAVE THE MEMORY ADDRESS
76355  122403           ST      X2,FIRST,X1
76356    5636           JMP     RESTOR          RESTORE REGISTERS

76357   70402  LFTERM   CPI     1,2
76360    5010           BNZ     ARROWTERM
76361   32401           ADI     X2,1            LF, OPEN NEXT MEM LOC
76362   34015           LI      0,X'0D'
76363    2340           JSR     *PUTC1          OUTPUT CR
76364  171405  MORMEM   RMOV    3,X2
76365    2341           JSR     *BINHX1         OUTPUT THE ADDRESS
76366   34057           LI      0,X'2F'
76367    2334           JSR     *PUTC1          OUTPUT A SLASH
76370    5756           JMP     NXTMEM

76371   70403  ARROWTERM CPI    1,3
76372    1332           BNZ     *QUESTN1        UNEXPECTED TERMINATOR
76373   52777           ADI     X2,-1           UP ARROW, DECREMENT PNTR
76374    2351           JSR     *CRLF1          OUTPUT CR & LF
76375    5766           JMP     MORMEM
```

```
                        *
                        *       OPENCUR DISPLAYS MEM LOC PNTED TO BY FIRST, AND WAITS FOR
                        *       HEX-INPUT AS IN OPENMEM.
                        *
76376  166403  OPENCUR  LD      X2,FIRST,X2
76377    2326           JSR     *CRLF1
76400    5763           JMP     MORMEM
                        *
                        *       OPENXT INCREMENTS FIRST, DISPLAYS MEM LOC PNTED TO BY IT,
                        *       AND WAITS FOR HEX INPUT AS IN OPENMEM.
                        *
76401  165003  OPENXT   LD      2,FIRST,X2
76402   31001           ADI     2,1
76403  161003           ST      2,FIRST,X2
76404  172402           RMOV    X2,2
76405    5756           JMP     MORMEM
                        *
                        *       OPENPREV DECREMENTS FIRST, DISPLAYS MEM LOC PNTED TO BY IT,
                        *       AND WAITS FOR HEX INPUT AS IN OPENMEM.
                        *
76406  165003  OPENPREV LD      2,FIRST,X2
76407   31377           ADI     2,-1
76410  161003           ST      2,FIRST,X2
76411  172402           RMOV    X2,2
76412    5751           JMP     MORMEM
                        *
                        *       OPENLOC DISPLAYS LOC AND WAITS FOR HEX-INPUT
                        *
76413   34040  OPENLOC  LI      0,X'20'
76414    2307           JSR     *PUTC1          OUTPUT *SPACE
76415  165406           LD      1,LOC,X2
76416    2310           JSR     *BINHX1         OUTPUT LOC CONTENTS

76417   32406           ADI     X2,LOC          SET POINTER TO LOC
76420    2307           JSR     *GETHX1         GET & STORE HEX CHARACTERS

76421   70401           CPI     1,1             CHECK FOR CR TERMINATOR
76422    1302           BNZ     *QUESTN1
76423    1443           JMP     *RESTADDR
                        *
                        *       SAVFRST  SAVES R2 (1ST ADDR TO BE OPENED/PUNCHED/COPIED) IN FI
                        *
76424  161003  SAVFRST  ST      2,FIRST,X2
76425   35000           LI      2,0
76426  161007           ST      2,CONVERT,X2
76427    1701           JMP     *INTRP1
                        *
                        *       SAV2ND SAVES R2 (LAST ADDR TO BE COPIED) IN SECOND.
                        *
76430  161004  SAV2ND   ST      2,SECOND,X2
76431   35000           LI      2,0
76432  161007           ST      2,CONVERT,X2
76433    1675           JMP     *INTRP1
```

```
476                     *
477                     *       EXECUTE RESTORES ALL THE TRAPS, CLEARS THE CURRENT TRAP FROM TH
478                     *       STACK IF CURTRAP >-1, SETS CURTRAP =-1, RESTORES THE USER'S
479                     *       REGISTERS, AND BEGINS EXECUTION OF THE USER'S PROGRAM AT THE LO
480                     *       ATION POINTED TO BY R2 IF HEX DIGITS WERE ENTERED OR LOC OTHERW
481                     *
482  76434   34377  EXECUTE LI      0,-1
483  76435  150030          ST      0,CURTRAP,X2    RESET CURTRAP
484  76436    2267          JSR     *CNLP1

486  76437  164006          LD      0,LOC,X2
487  76440  164407          LD      1,CONVERT,X2    CHECK TO SEE IF HEX DIGITS WERE INPUT
488  76441    4401          BZ      $+2             NO
489  76442  170002          RMOV    0,2             YES, USE CONTENTS OF R2 AS STARTING AD
490  76443  167016          LD      6,REG6,X2       RESTORE USER'S STACK POINTER
491  76444  170166          PUSH    SP,0            PUSH STARTING ADDR ON STACK
492  76445   34000          LI      0,0
493  76446  160007          ST      0,CONVERT,X2
494  76447    6002          JSR     RSTRTRAP        RESTORE THE TRAPS
495  76450    2013          JSR     *RSTREG         RESTORE USER'S REGISTERS
496  76451  174120          RTS                     GO TO USER'S PROG

498  76452   34774  RSTRTRAP LI      1,-4
499  76453  134011          LDR     0,TRPINSTR      GET TRAP
500  76454  164000  NXTRP   LD      X1,TRAPADDR,X2  GET THE ADDR
501  76455  125000          LD      2,0,X1          FETCH USER'S INSTRUC IN CASE IT WAS CH
502  76456  161024          ST      2,TRAPINST,X2   SAVE IT
503  76457  120000          ST      0,0,X1          INSERT TRAP IN USER'S PROG
504  76460   32401          ADI     X2,1
505  76461   74772          BINC    1,NXTRP
506  76462   32774          ADI     X2,-4           RESTORE X2
507  76463  174120          RTS

509  76464   76700  RSTREG  DATA    RESREG
510  76465  173774  TRPINSTR IOCP   TRAP
511  76466    1640  RAMPIA  DATA    VARADDR
512  76467   76215  RESTADDR DATA   RESTOR
```

```
514                     *
515                     *       EXECNST RETURNS CONTROL TO THE USER'S PROGRAM AT THE LAST ENCOU
516                     *       BREAKPOINT.  A NUMBER OF ITEMS ARE OF IMPORTANCE BECAUSE OF THE
517                     *       IN WHICH THIS CONTROL MUST BE RETURNED (THE 1ST INSTRUC IS INTE
518                     *       IN A WORK AREA SEPARATE FROM THE USER'S PROG):
519                     *       1.  THE CONDITION CODE IS ALTERED BY EVERY INSTRUCTION,
520                     *       2.  INSTRUCTIONS USING RELATIVE ADDRESSING MUST BE RECOGNIZED.
521                     *
522                     *       TO CONTINUE THE USER'S PROGRAM FROM A TRAP, DEBUG MUST:
523                     *       INTERPRET THE INSTRUCTION ON WHICH THE TRAP HAD BEEN PLACED
524                     *       IF THE INSTRUC DOES NOT USE RELATIVE ADDRESSING -
525                     *       CHANGE THE TRAP INTERRUPT ADDRESS SO A 2ND INTERRUPT MAY BE
526                     *       TRIGGERED IN DEBUG IN ORDER TO PRESERVE THE USER'S STATUS
527                     *       BEFORE RETURNING TO THE USER'S PROG,
528                     *       EXECUTE THE USER'S INSTRUCTION IN A SPECIAL WORK AREA IN RAM,
529                     *       TRIGGER AN INTERRUPT,
530                     *       ON RETURN FROM THIS INTERRUPT,
531                     *       THE DMA STATUS DATA IS REMOVED,
532                     *       THE INTERRUPT PC DATA IS REPLACED BY THE USER'S PC,
533                     *       AND AN RTI IS EXECUTED.
534                     *
535                     *       AN RTI EXPECTS THE FOLLOWING DATA TO BE ON THE STACK:
536                     *
537                     *       TOP OF STACK -->        LEVEL
538                     *                               PC
539                     *                               STATUS
540                     *
541                     *       THE FOLLOWING INSTRUCTIONS USE RELATIVE ADDRESSING & THEREFORE
542                     *       HAVE THEIR EFFECTIVE ADDRESS ADJUSTED TO POINT TO THE PROPER LO
543                     *       LDR*, STR*, & LD PERFORM THE SAME INTERRUPT TRIGGERING OPERATIO
544                     *       JMP'S & JSR'S ARE INTERPRETED DIRECTLY
545                     *
546  76470  165430  EXECNXT  LD      1,CURTRAP,X2    MAKE SURE TRAP OCCURRED (CURTRAP > -1)
547  76471    5E33          BN      *QUESTN1
548  76472    2233          JSR     *CRLF1
549  76473    6356          JSR     RSTRTRAP        RESTORE TRAPS IN USER'S PROGRAM
550  76474  167016          LD      6,REG6,X2       RESTORE USER'S STACK POINTER

552  76475  172005          RMOV    X1,X2
553  76476  164030          LD      0,CURTRAP,X2    COMPUTE PNTR
554  76477  172020          RADD    X1,0            TO TRAP ENTRY
555  76500  124024          LD      0,TRAPINST,X1   GET USER'S INSTRUC
556  76501  160031          ST      0,USRINSTR,X2   PUT IT IN WORK AREA
557  76502  170400          RMOV    1,0             SAVE IT FOR LATER
558  76503   35777          LI      3,-1
559  76504  161430          ST      3,CURTRAP,X2    RESET CURTRAP TO -1

561  76505  170255          SHR     0,12            ISOLATE MS 4 BITS OF INSTR TO EXAMINE
562  76506   70000          CPI     0,0             CHECK FOR JMP, BRNCH, RTI, OR IMSK
563  76507    4465          BZ      JMPBRNCH
564  76510   70002          CPI     0,2             CHECK FOR LDR* OR STR*
565  76511    4441          BZ      LDSTRIND
566  76512   70013          CPI     0,X'B'          CHECK FOR LD OR LIM
567  76513    4455          BZ      LDLIMREL
568  76514   70007          CPI     0,7             CHECK FOR BINC
569  76515     546          BZ      *BINCADDR
570  76516   70017          CPI     0,X'F'          CHECK FOR SNR OR RTS
571  76517     545          BZ      *SKRTSAD
572  76520   70001          CPI     0,1             CHECK FOR SKAZ OR LDCK
573  76521     548          BZ      *SKZLOKAD
```

```
576  76522  134131  SETRAP  LDR     0,TPADR2     MODIFY TRAP ADDR SO 2ND
577  76523   60004          ST      0,TPINT      INTERRUPT GETS BACK INTO DEBUG

579  76524  134340          LDR     0,TRPINSTR
580  76525  160032          ST      0,STATINST,X2  INSERT 2ND TRAP IN WORK AREA
581  76526    6151          JSR     RESREG       RESTORE REGISTERS
582  76527    1522          JMP     *SIMULAT

584  76530  172566  STATRTRN PUSH   SP,X2        2ND INTERRUPT WILL RETURN HERE
585  76531  136734          LDR     X2,RAMP1A    GET
586  76532  160010          ST      0,REG0,X2    WORKING
587  76533  170206          POP     SP,0         ROOM FOR
588  76534  160015          ST      0,REG5,X2    REGISTERS

590  76535  170206          POP     SP,0         REMOVE LEVEL
591  76536  170206          POP     SP,0         REMOVE PSEUDO PC
592  76537  170206          POP     SP,0         REMOVE 2 WDS
593  76540  170206          POP     SP,0         OF DMA STATUS
594  76541  164040          LD      0,USRPC,X2
595  76542  170166          PUSH    SP,0         PUSH PC
596  76543  164041          LD      0,USRLEVEL,X2
597  76544  170166          PUSH    SP,0         PUSH LEVEL

599  76545  134105          LDR     0,TPADR1
600  76546   60004          ST      0,TPINT      RESTORE NORMAL TRAP INTERRUPT ADDRESS

602  76547  164010          LD      0,REG0,X2    RESTORE USER'S
603  76550  166415          LD      5,REG5,X2    REGISTERS
604  76551  173777          IOCP    CLRTRAP      CLEAR TRAP INTERRUPT
605  76552    7400          RTI                  GO TO USER'S PROG

607  76553  174647  LDSTRIND ROT    1,8          R1 CONTAINED USER'S INSTRUCTION
608  76554  170647          SHRA    1,8          SIGN EXTEND THE DISPLACEMENT
609  76555  142440          ADD     1,USRPC,X2   ADD DISPLACEMT TO PC TO GET EFFEC ADDR
610  76556  172001          RMOV    X1,1
611  76557  124400          LD      1,0,X1
612  76560  160433  SETINSTR ST     1,USROPRND,X2
613  76561  164431          LD      1,USRINSTR,X2  GET USER'S INSTRUC
614  76562  135076          LDR     2,INSTMSK
615  76563  170462          RAND    1,2
616  76564   30401          ADI     1,1          MAKE THE DISPL = $+2
617  76565  160431          ST      1,USRINSTR,X2  PUT IT IN THE WORK AREA
618  76566    5733          JMP     SETRAP       SETUP 2ND TRAP

620  76567  174647  LDLIMREL ROT    1,8
621  76570  170647          SHRA    1,8          SIGN EXTEND DISPL
622  76571  156040          LD      X1,USRPC,X2
623  76572  172021          RADD    X1,1         COMPUTE EFFECTIVE ADDR
624  76573  124400          LD      1,0,X1       GET USER'S DATA
625  76574    5763          JMP     SETINSTR
```

```
628  76575  34007  JMPBRNCH LI    0,7
629  76576  174623         SHL     1,4
630  76577  170633         SHR     1,12          ISOLATE 2NDARY OPCODE
631  76600  170061         RAND    2,1           ALSO TAKE LS 3 BITS
632  76601  70417          CPI     1,X'F'        CHECK FOR RTI
633  76602  4471           BZ      RTNCNTRL
634  76603  70407          CPI     1,7           CHECK FOR IMSK
635  76604  4441           BZ      IMSKINST
636  76605  70410          CPI     1,8           CHECK FOR BLOC
637  76606  4713           BZ      SETRAP
638  76607  70400          CPI     1,0           CHECK FOR JSRZ
639  76610  4476           BZ      JSRZINST
640  76611  70003          CPI     0,3           CHECK FOR JMP
641  76612  4503           BZ      JMPINST
642  76613  70004          CPI     0,4           CHECK FOR JSR
643  76614  4512           BZ      JSRINST

645                  *
646                  *       CONDITIONAL BRANCHES ARE INTERPRETED BY USING THE INTERRUTP
647                  *       STATUS IN THE STACK, EXECUTING THE FOLLOWING INSTRUC IN
648                  *       THE WORK AREA:
649                  *       LD 0,$+3 (USRINSTR)
650                  *       CONDITIONAL BRANCH W/ DISPL = *$+3 (STATINST)
651                  *       JMP *$+3 (USROPRND)
652                  *       INTERRUPTED STATUS (SCRCH1)
653                  *       ADDR IN DEBUG FOR SUCCESSFUL TEST (SCRCH2)
654                  *       ADDR IN DEBUG FOR UNSUCCESSFUL TEST (SCRCH3)
655                  *
656  76615  164431         LD      1,USRINSTR,X2 NONE OF ABOVE, MUST BE CONDITIONAL BRA
657  76616  171001         RMOV    2,1
658  76617  134035         LDR     0,LDSTAT      SET UP LD STATUS INSTRUC
659  76620  160031         ST      0,USRINSTR,X2

661  76621  135440         LDR     3,BRNCHMSK    SET UP CONDITIONAL BRANCH, GET INSTR M
662  76622  171461         RAND    3,1           MAKE IT INDIRECT
663  76623  31402          ADI     3,2           MAKE IT REL $+3
664  76624  161432         ST      3,STATINST,X2 INSERT CONDITIONAL BRANCH IN WORK AREA
665  76625  174627         SHL     1,8           SIGN EXTEND DISPLACEMENT
666  76626  170647         SHRA    1,8
667  76627  142440         ADD     1,USRPC,X2    COMPUTE SUCCESSFUL BRANCH ADD
668  76630  172001         RMOV    X1,1
669  76631  175223         SHL     2,4           CHECK FOR
670  76632  7001           BN      $+2           INDIRECT ADDRESSING
671  76633  124400         LD      1,0,X1        IT WAS
672  76634  160442         ST      1,$BA,X2      INSERT IN WORK AREA
673  76635  134020         LDR     0,JSTRUC      SET UP JMP INSTRUC
674  76636  160234         ST      0,USROPRND,X2
675  76637  164037         LD      0,USRSTAT,X2  SET UP STATUS DATA
676  76640  160034         ST      0,SCRCH1,X2
677  76641  134015  SETRTRNS LDR    0,RETRN1      SET UP RETURN ADDRESSES
678  76642  160035         ST      0,SCRCH2,X2
679  76643  134014         LDR     0,RETRN2
680  76644  160035         ST      0,SCRCH2,X2
681  76645  1404           JMP     *SIMULAT
```

```
 684   76646   165431  IMSKINST LD      3,USRINSTR,X2
 685   76647   161441           ST      3,USRLEVEL,X2
 686   76650   164040           LD      0,USRPC,X2
 687   76651     5416           JMP     SETSTACK

 689   76652     1671  SIMULAT  DATA    USRINSTR+VARADDR
 690   76653    77073  TPADR1   DATA    TRAPRTRN        NORMAL TRAP ADDRESS
 691   76654    76530  TPADR2   DATA    STATRTRN        ALTERNATE TRAP ADDRESS
 692   76655   134002  LDSTAT   LDR     0,S+3
 693   76656     1402  JSTRUC   JMP     *S+3
 694   76657    76667  RETRN1   DATA    SUCCESS
 695   76660    76675  RETRN2   DATA    NOSUCCES
 696   76661   177400  INSTMSK  DATA    X'FF00'
 697   76662     3400  BRNCHMSK DATA    X'0700'
 698   76663      377  DISPMSK  DATA    X'00FF'
 699   76664    76742  BINCADDR DATA    BINCINST        SA
 700   76665    76770  SKRTSAD  DATA    SKRRTS          ABSA
 701   76666    77003  SKZLGRAD DATA    SKRAZLGR        ABSA
 702              *

 704   76667   164042  SUCCESS  LD      0,6BA,X2        CONDITIONAL TEST SUCCEEDED
 705   76670   170166  SETSTACK PUSH    SP,0            PUSH DUMMY STATUS
 706   76671   170166           PUSH    SP,0            PUSH RETURN ADDR
 707   76672   164041           LD      0,USRLEVEL,X2
 708   76673   170166           PUSH    SP,0            PUSH INTERRUPT LEVEL
 709   76674     6003  RTNCNTRL JSR     RESREG
 710   76675     7400           RTI

 712   76676   164040  NOSUCCES LD      0,USRPC,X2      CONDITIONAL TEST FAILED
 713   76677     5770           JMP     SETSTACK

 715   76700   164010  RESREG   LD      0,REG0,X2
 716   76701   164411           LD      1,REG1,X2
 717   76702   165012           LD      2,REG2,X2
 718   76703   165413           LD      3,REG3,X2
 719   76704   166014           LD      4,REG4,X2
 720   76705   166415           LD      5,REG5,X2
 721   76706   174120           RTS

 723   76707   164040  JSRZINST LD      0,USRPC,X2
 724   76710   170166           PUSH    SP,0            PUSH RETURN ADDR ON STACK
 725   76711   164031           LD      0,USRINSTR,X2
 726   76712   136550           LDR     X1,DISPMSK
 727   76713   172060           RAND    X1,0
 728   76714   124000           LD      0,0,X1          GET *PAGE ZERO ADDR
 729   76715     5752           JMP     SETSTACK        SET UP RTI
```

```
732  76716  164041  JMRINST  LD      0,USRINSTR,X2
733  76717  174227           SHL     0,8
734  76720  170247           SHRA    0,8          SIGN EXTEND DISPLACEMENT
735  76721  142040           ADD     0,USRPC,X2
736  76722  172000           RMOV    X1,0
737  76723   70403           CPI     1,5          CHECK FOR INDIRECT
738  76724    5001           BNZ     $+2          NO
739  76725  124000           LD      0,0,X1       YES, FETCH ADDRESS
740  76726    5741           JMP     SETSTACK     SET UP RTI

742  76727  164040  JSRINST  LD      0,USRPC,X2
743  76730  170164           PUSH    SP,0         PUSH RETURN ADDR ON STACK
744  76731  165041           LD      2,USRINSTR,X2
745  76732  175227           SHL     2,8
746  76733  171247           SHRA    2,8          SIGN EXTEND DESPLACEMENT
747  76734  170022           RADD    2,2
748  76735  172000           RMOV    X1,0
749  76736   70404           CPI     1,4          CHECK FOR INDIRECT
750  76737    5001           BNZ     $+2          NO
751  76740  124000           LD      0,0,X1       IS, FETCH ADDR
752  76741    5726           JMP     SETSTACK     SET UP RTI
753                  *
754                  *       FOR A BINC, THE FOLLOWING INSTRUCS ARE EXECUTED IN THE WORK ARE
755                  *       BINC    RX,$+2       (USRINSTR)
756                  *       JMP     *$+3         (STATINST)
757                  *       JMP     *$+1         (USROPRND)
758                  *       ADDR IN DEBUG FOR RX NOT = 0  (SCRCH1)
759                  *       ADDR IN DEBUG FOR RX = 0 (SCRCH2)
760                  *
761  76742   34100  BINCINST LI      0,X'40'      MAKE
762  76743  174224           SHL     0,5          SURE
763  76744  170061           RAND    0,1          IT'S NOT
764  76745     420           BZ      *SETRPAD     CPI

765  76746  174627           SHL     1,8
767  76747  170647           SHRA    1,8          SIGN EXTEND DISPLACEMENT
768  76750  142440           ADD     1,USRPC,X2   COMPUTE USER'S LOOP ADDRESS
769  76751  160442           ST      1,SRA,X2

771  76752  134303           LDR     0,JSTRUC     SET UP NOT ZERO BRANCHINSTRUC
772  76753  160032           ST      0,STATINST,X2
773  76754  134010           LDR     0,JINSTRC    SET UP ZERO BRANCH INSTRUC
774  76755  160033           ST      0,USROPRND,X2

776  76756  124024           LD      0,TRAPINST,X1  SET UP BINC INSTRUCTION IN WORK AREA
777  76757  134701           LDR     1,INSTMSK
778  76760  170460           RAND    1,0          MASK OUT DISPLACEMENT
779  76761   50401           ADI     1,1          MAKE DISPLACEMENT = $+2
780  76762  160431           ST      1,USRINSTR,X2  INSERT IN WORK AREA

782  76763    6314           JSR     RESREG
783  76764    1402           JMP     *SUBRTRN
```

```
                        ***
                        *
76765    1400   JINSTRC  JMP       *$+1
76766   76522   SETRPAD  DATA      SETRAP          ABSA
76767   76641   SUDORTRN DATA      SETRTRNS        ABSA
                        *
                        ***
                        *
                        *       WORK AREA FOR THE SKIP INSTRUCTIONS:
                        *       THE SKIP INSTRUCTION     (USRINSTR)
                        *       JMP       *$+4           (STATINST)
                        *       JMP       *$+2           (USROPRND)
                        *                                (SCRCH1)
                        *       ADDR IN DEBUG FOR SUCCESSFUL TEST    (SCRCH2)
                        *       ADDR IN DEBUG FOR UNSUCCESSFUL TEST  (SCRCH3)
                        *
76770  174623   SKRRTS   SHL       1,4
76771    2774           BP        *SETRPAD
76772  170627           SHR       1,8             ISOLATE SECONDARY OP CODE
76773   34017           LI        0,X'F'
76774  170061           RAND      0,1
76775   70017           CPI       0,X'F'          CHECK FOR SKR
76776    4404           BZ        SKRAZLOK
76777   70005           CPI       0,5             CHECK FOR RTS
77000    1365           BNZ       *SETRPAD
77001  170206           POP       SP,0
77002    5665           JMP       SETSTACK

77003  134361   SKRAZLOK LDR       0,JINSTRC       SKR, SKAZ OR LOCK
77004   30001           ADI       0,1
77005  160033           ST        0,USROPRND,X2
77006   30002           ADI       0,2
77007  160032           ST        0,STATINST,X2
77010  164040           LD        0,USRPC,X2      SET UP ADDR FOR SUCCESSFUL TEST
77011   30001           ADI       0,1
77012  160042           ST        0,SRA,X2
77013    1753           JMP       *SUDORTRN
                        *
                        *
                        *
                        ******* EXECNXT ENDS HERE  ************************
```

```
                        *
                        *       DISTRAP CHECKS CONVERT FLAG TO DETERMINE WHETHER A SINGLE TRAP
                        *       (POINTED TO BY R2) IS TO BE OPENED OR ALL TRAPS ARE TO BE
                        *       DISPLAYED.  THEN CARRIES OUT THE REQUIRED OPERATION.
                        *
77014  164007  DISPTRAP  LD      0,CONVERT,X2
77015    4425           BZ      ALLTRAPS        IF NO HEX DIGIT ENTERED, DISP ALL TRAP
77016    6145           JSR     CHKTRAP         MAKE SURE REG 2 CONTAINS VALID TRAP #

77017  34040            LI      0,X'20'
77020    2163           JSR     *PUTC2          OUTPUT *SPACE
77021  172022           RADD    X2,2            COMPUTE PNTR TO TRAP IN RAM AREA
77022   52420           ADI     X2,TRAPADDR     TO BE USED BY GETHEX
77023  165400           LD      3,0,X2
77024  171566           PUSH    SP,3            SAVE ORIGINAL TRAP ADDR.
77025  134153           LDR     0,MULTRPAD
77026  170123           RCMP    0,3             CHECK FOR A NONEXISTENT TRAP
77027    4401           BZ      $+2
77030    2156           JSR     *BINHX2         OUTPUT TRAP PNTED TO BY R2
77031    2155           JSR     *BINHX2
77032   70401           CPI     1,1             CHECK FOR CR TERMINATOR
77033    5034           BNZ     BADCHAR
77034  172206           POP     SP,X1           SEE IF TRAP MODIFIED
77035  171124           RCMP    2,X1            SEE IF TRAP MODIFIED
77036    4533           BZ      RESET           IF NOT, FINISHED
77037  172002           RMOV    X1,2            ELSE SAVE NEW INSTRUC, R2 = NEW TRAP A
77040  124000           LD      0,0,X1          GET NEW INSTR
77041  160004           ST      0,TINSTPTR,X2   SAVE IT
77042    5527           JMP     RESET

77043  35374  ALLTRAPS  LI      2,-4            R2 WILL = LOOP CNTR
77044  32420            ADI     X2,TRAPADDR     COMPUTE RAM PTR TO TRAPS
77045    5402           JMP     $+3             OMITT 1ST TIME THRU LOOP
77046  34124  TRAPLOOP  LI      0,X'54'
77047    2134           JSR     *PUTC2          OUTPUT T
77050  34040            LI      0,X'20'
77051  170022           RADD    0,2             TRAP#
77052    2131           JSR     *PUTC2          OUTPUT *SPACE
77053  34040            LI      0,X'20'
77054    2127           JSR     *PUTC2          OUTPUT *SPACE
77055  165400           LD      3,0,X2
77056  134122           LDR     0,MULTRPAD
77057  170123           RCMP    0,3             CHECK FOR A NONEXISTENT TRAP
77060    4401           BZ      $+2
77061    2125           JSR     *BINHX2         OUTPUT TRAP ADDRESS
77062  34040            LI      0,X'20'
77063    2120           JSR     *PUTC2          OUTPUT
77064    2117           JSR     *PUTC2          2 *SPACES
77065  32401            ADI     X2,1
77066  75357            BINC    2,TRAPLOOP
77067    5502           JMP     RESET

77070  171604  BADCHAR  POP     SP,3            RETRIEVE OLD TRAP ADDRESS
77071  161400           ST      3,0,X2          RESTORE OLD TRAP ADDR
77072    1512           JMP     *QUEST2
         4    TINSTPTR  EQU     TRAPINST-TRAPADDR
```

```
885                         *
886                         *       TRAPRTN SAVES THE USER'S REGISTERS AND SEARCHES THE TRAP TABLE
887                         *       TO DETERMINE WHICH TRAP WAS ENCOUNTERED.  IT THEN OUTPUTS THE
888                         *       TRAP # AND THE ADDRESS OF THE TRAP AND GOES TO THE INTERPRETER
889                         *       TO WAIT FOR A COMMAND FROM THE KEYBOARD.
890                         *
891   77073  172566  TRAPRTRN PUSH    SP,X2
892   77074  136505           LDR     X2,RAMPT2       GET ADDRESS OF RAM
893   77075  160010           ST      0,REG0,X2       **
894   77076  160411           ST      1,REG1,X2       *
895   77077  161012           ST      2,REG2,X2       * SAVE
896   77100  161413           ST      3,REG3,X2       * USER'S
897   77101  162014           ST      4,REG4,X2       * REGISTERS
898   77102  170006           POP     SP,0            *
899   77103  160015           ST      0,REG5,X2       **

901   77104  173777           IOCP    CLRTRAP         CLEAR INTERRUPT
902   77105    3403           IMSK    3               REENABLE TRAP
903   77106  171206           POP     SP,2
904   77107  161041           ST      2,USRLEVEL,X2   SAVE LEVEL
905   77110  171606           POP     SP,3
906   77111  161C 3           ST      3,USRPC,X2      SAVE PC
907   77112  170006           POP     SP,0            DELETE DMA ENTRIES
908   77113  170206           POP     SP,0            FROM STACK
909   77114  170206           POP     SP,0
910   77115  164037           ST      0,USRSTAT,X2    SAVE STATUS
911   77116  165016           ST      6,REG6,X2

913   77117   31777           ADI     3,-1
914   77120  161417           ST      3,REG7,X2
915   77121  172025           RMOV    X1,X2           X1 WILL POINT TO PROPER TRAP ENTRY
916   77122  115420  NXTRAP   CMPR    3,TRAPADDR,X1   SEARCH FOR PC IN TRAP TABLE
917   77123    4402           BZ      FOUNDTRP        TO GET THE TRAP NUMBER
918   77124   32001           ADI     X1,1
919   77125    5774           JMP     NXTRAP

921   77126  172045  FOUNDTRP RSUB    X1,X2           COMPUTE TRAP NUMBER
922   77127  162030           ST      X1,CURTRAP,X2
923   77130   30124           LI      0,X'54'
924   77131    2052           JSR     *PUTC2          OUTPUT T
925   77132   30060           LI      0,X'30'
926   77133  172024           RADD    0,X1            COMPUTE ASCII CHAR FOR TRAP#
927   77134    2047           JSR     *PUTC2          OUTPUT TRAP #
928   77135   30040           LI      0,X'20'
929   77136    2045           JSR     *PUTC2          OUTPUT *SPACE
930   77137    6124           JSR     BINHEX          OUTPUT TRAP ADDRESS
931   77140   30007           LI      2,7
932   77141    2042           JSR     *PUTC2          OUTPUT DING

934   77142   30774           LI      1,-4            RESTORE
935   77143  166020  RSTRINST LD      X1,TRAPADDR,X2,TO
936   77144  120000           ST      0,0,X1          USER'S
937   77145   32401           ADI     X2,1            PROGRAM
938   77146   74774           BINC    1,RSTRINST
939   77147    5422           JMP     RESET
```

```
941                     *
942                     *       REMOVTRP USES THE CONVERT FALG TO DETERMINE WHETHER A SINGLE
943                     *       TRAP (POINTED TO BY R2) IS TO BE REMOVED OR ALL TRAPS ARE TO
944                     *       BE DELETED.
945                     *
946  77150  130030  REMOVTRP LDR     0,NULTRPAD
947  77151  164007          LD      1,CONVERT,X2
948  77152    4404          BZ      DELETALL        IF NO HEX DIGIT ENTERED, DELETE ALL TR
949  77153    6010          JSR     CHKTRAP         MAKE SURE REG 2 CONTAINS VALID TRAP #

951  77154  172422          RADD    X2,2            COMPUTE PNTR TO TRAP IN RAM AREA
952  77155  160020          ST      0,TRAPADDR,X2   REMOVE IT FROM THE TABLE
953  77156    5413          JMP     RESET

955  77157   35374  DELETALL LI      2,-4
956  77160  160020          ST      0,TRAPADDR,X2
957  77161   32401          ADI     X2,1
958  77162   75375          BINC    2,DELETALL+1
959  77163    5406          JMP     RESET

961  77164   71000  CHKTRAP  CPI     2,0             MAKE SURE REG 2 CONTAINS VALID TRAP #
962  77165    6401          BP      $+2
963  77166    1416          JMP     *QUEST2         LESS THAN 0
964  77167   71004          CPI     2,4
965  77170    2414          BP      *QUEST2         GREATER THAN 3
966  77171  174120          RTS

968  77172   35000  RESET    LI      2,0
969  77173   35400          LI      3,0
970  77174  136405          LDR     X2,RAMPT2
971  77175  161007          ST      2,CONVERT,X2
972  77176  161002          ST      2,HXARITH,X2
973  77177    2006          JSR     *CRLF2
974  77200    1410          JMP     *INTRP2

976  77201   76151  NULTRPAD DATA    NULTRAP         ADDRESS OF NULL TRAPS
977  77202    1640  RAMPT2   DATA    VARADDR
978  77203   77365  GETC2    DATA    GETC            ***NOT REQUIRED***
979  77204   77407  PUTC2    DATA    PUTC
980  77205   77354  QUEST2   DATA    QUESTION
981  77206   77360  CRLF2    DATA    CRLF
982  77207   77266  BINHX2   DATA    BINHEX
983  77210   77303  GETHX2   DATA    GETHEX
984  77211   76120  INTRP2   DATA    INTERPRT
```

```
                        *
                        *       HEX ARITHMETIC ROUTINES
                        *
                        *       HXARITH IS USED AS A COMMUNICATION FLAG BETW THESE ROUTINES
                        *       AND TAKES ON THE FOLLOWING VALUES
                        *       -1 IF '-' OPERATOR WAS LAST RECEIVED
                        *       0 IF NO PERATOR HAS BEEN RCVD FOR CURRENT EXPRESSION
                        *       +1 IF '+' OPERATOR WAS LAST RECEIVED
                        *
                        *       THE EXPRESSION IS EVALUATED AS OPERANDS & OPERATORS ARE
                        *       RECEIVED,  PARENTHETICAL EXPRESSIONS ARE NOT RECOGNIZED.
                        *
                        *       REGISTER 3 CONTAINS THE CURRENT EVALUATION OF THE EXPRESSION
                        *       REGISTER 2 CONTAINS THE CURRENT OPERHAND
                        *
1001  77212    6016     HXADD   JSR     PREVOP
1002  77213   34401             LI      1,1
1003  77214  160400             ST      1,HXARITH,X2    HXARITH = 1 FOR ADD
1004  77215    1773             JMP     *INTRP2
1005                    *
1006                    *
1007  77216    6012     HXSUB   JSR     PREVOP
1008  77217   34777             LI      1,-1
1009  77220  160400             ST      1,HXARITH,X2    HXARITH = -1 FOR SUB
1010  77221    1767             JMP     *INTRP2
1011                    *
1012                    *
1013  77222    6006     HXRESULT JSR    PREVOP
1014  77223    6042             JSR     BINHEX          OUTPUT CURRENT EVALUATION
1015  77224   35000             LI      2,0
1016  77225    6132             JSR     CRLF            OUTPUT CR & LF
1017  77226  161000             ST      2,HXARITH,X2    RESET OPERATION FLAG (HXARITH=0)
1018  77227   35400             LI      3,0             CLEAR CURRENT SUM
1019  77230    1760             JMP     *INTRP2
1020                    *
1021                    *
1022  77231   34400     PREVOP  LI      1,0
1023  77232  160407             ST      1,CONVERT,X2    CLEAR CONVERT FLAG
1024  77233  164400             LD      1,HXARITH,X2
1025  77234    4405             BZ      NOPREV          CHECK FOR PREVIOUS OPERATOR
1026  77235   72400             CPI     1,0             CHECK FOR '-' OPERATOR
1027  77236    7006             BN      SUBPREV
1028  77237  171422             RADD    3,2             NOPE, MUST BE '+' OPERATOR
1029  77240   35000             LI      2,0
1030  77241  174120             RTS
1031  77242  171402     NOPREV  RMOV    3,2             SAVE OPERAND
1032  77243   35000             LI      2,0
1033  77244  174120             RTS
1034  77245  171442     SUBPREV RSUB    3,2
1035  77246   35000             LI      2,0
1036  77247  174120             RTS
1037  77250    7777     XFFF    DATA    X'FFF'
1038  77251   10000     FOURK   DATA    4096
```

```
1040                    *
1041                    *       HEXALPH CONVERTS ALPHA DATA IN R0 AND MERGES IT INTO R2
1042                    *
1043                    *       CONVERT IS A FLAG USED BY ROUTINES THAT ARE INVOKED BY CERTAIN
1044                    *       DEBUG COMMANDS.  THESE ROUTINES MUST DETERMINE IF THE COMMAND
1045                    *       HAS BEEN PRECEDED BY A HEXCHARACTER(S) (EG. 1358/ FOR OPEINING
1046                    *       A MEMORY LOC OR 4R FOR OPENING REGISTER 4) AND FOR THIS PURPOSE
1047                    *       CONVERT IS SET TO 1 IF HEX INPUT HAS BEEN RECEIVED.
1048                    *
1049  77252   30364     HEXALPH ADI     0,-12
1050  77253   175223            SHL     2,4             SHIFT CURRENT CONTENTS OF R2
1051  77254   171020            RADD    2,0             INSERT DATA FROM R0
1052  77255   34401             LI      1,1
1053  77256   160407            ST      1,CONVERT,X2    SET CONVERT FLAG
1054  77257    1751             JMP     *INTRP2

1056                    *
1057                    *       HEXNUM CONVERTS NUMERIC DATA IN R0 AND MERGES IT INTO R2
1058                    *
1059  77260   30373     HEXNUM  ADI     0,-5
1060  77261   175223            SHL     2,4             SHIFT CURRENT CONTENTS OF R2
1061  77262   171020            RADD    2,0             INSERT DATA FROM R0
1062  77263   34401             LI      1,1
1063  77264   160407            ST      1,CONVERT,X2    SET CONVERT FLAG
1064  77265    1723             JMP     *INTRP2

1066                    *
1067                    *       BINHEX CONVERTS NUMBER IN R3 TO HEX AND OUTPUTS IT
1068                    *
1069  77266   170566    BINHEX  PUSH    SP,1
1070  77267   34774             LI      1,
1071  77270   34017     NXTDIGIT LI     0,
1072  77271   115645           RDT
1073  77272   170063           RAND                    ISOLATE 1 DIGIT
1074  77273   30060            ADI     0,X
1075  77274   70072            CPI     0,X'3A'         CHECK IF > 9
1076  77275    7401            BN      $+2
1077  77276   30007            ADI     0,7             YES, CONVERT TO A...F
1078  77277    6107            JSR     PUTC            OUTPUT DIGIT
1079  77300   74767            BINC    1,NXTDIGIT
1080  77301   170006           POP     SP,1
1081  77302   174420           RTS
```

```
1083                    *
1084                    *       GETHEX GETS HES DIGITS FROM THE KEYBRD UNTIL A CR, LF, OR
1085                    *       UPARROW IS RECEIVED.  CONVERTS, STORES THEM IN LOC PNTED TO
1086                    *       BY X2, AND RETURNS A CODE IN R1 AS FOLLOWS:
1087                    *       0 - UNRECOGNIZABLE TERMINATOR
1088                    *       1 - CR TERMINATOR
1089                    *       2 - LF TERMINATOR
1090                    *       3 - UPARROW TERMINATOR
1091                    *
1092  77303  34040   GETHEX  LI      0,X'20'
1093  77304  6102           JSR     PUTC            OUTPUT *SPACE
1094  77305  34777          LI      1,-1            R1 WILL BE 'CHAR-RCVD' FLAG
1095  77306  35000          LI      2,0             R2 WILL ACCUMULATE HEX DIGITS
1096  77307  6055   NXTHEX  JSR     GETC            GET A CHAR
1097  77310  70015          CPI     0,X'0D'         CHECK FOR CR
1098  77311  5005           BNZ     CHKLF

1100  77312  74401          BINC    1,$+2           YES, CHECK IF HEX CHARS WERE INPUT
1101  77313  5401           JMP     $+2
1102  77314  161000         ST      2,0,X2          YES, STORE THEM
1103  77315  34401          LI      1,1             SET RETURN CODE TO 1
1104  77316  174120         RTS

1106  77317  70012  CHKLF   CPI     0,X'0A'         CHECK FOR LF
1107  77320  5005           BNZ     CHKARROW
1108  77321  74401          BINC    1,$+2           YES, CHECK IF HEX CHARS WERE INPUT
1109  77322  5401           JMP     $+2
1110  77323  161000         ST      2,0,X2          YES, STORE THEM
1111  77324  34402          LI      1,2             SET RETURN CODE TO 2
1112  77325  174120         RTS

1114  77326  70136  CHKARROW CPI    0,X'5E'         CHECK FOR UP ARROW
1115  77327  5005           BNZ     CNVRT
1116  77330  74401          BINC    1,$+2           YES, CHECK IF HEX CHARS WERE INPUT
1117  77331  5401           JMP     $+2
1118  77332  161000         ST      2,0,X2          YES, STORE THEM
1119  77333  34403          LI      1,3             SET RETURN CODE TO 3
1120  77334  174120         RTS

1122  77335  36042  CNVRT   LI      X1,X'30'
1123  77336  170044         RSUB    0,X1            TRY TO CONVERT TO BINARY
1124  77337  7006           BN      ILLEG
1125  77340  70012          CPI     0,10            CHECK IF LESS THAN 10
1126  77341  7006           BN      GOODHEX         IT IS, MERGE INTO CURRENT VALUE
1127  77342  30371          ADI     0,-7            ISNT, FURTHER CHECK REQUIRED
1128  77343  7002           BN      ILLEG           CHECK IF 1*(*)?*
1129  77344  70020          CPI     0,16            CHECK IF < 16
1130  77345  7002           BN      GOODHEX         IT IS

1132  77346  34400  ILLEG   LI      1,0             ILLEGAL CHAR, SET RETURN CODE TO 0
1133  77347  174120         RTS

1135  77350  174224 GOODHEX SHL     2,4             INSERT IT
1136  77351  171020         RADD    2,0
1137  77352  30401          ADI     1,1
1138  77353  5733           JMP     NXTHEX          GET ANOTHER CHAR
```

```
1140                        *
1141                        *       QUESTION IS ENTERED WHENEVER AN UNRECOGNIZABLE CONDITION EXISTS
1142                        *       SO THAT CERTAIN CRITICAL PARAMETERS MAY BE RESET
1143                        *
1144  77354   34077 QUESTION LI      0,X'3F'
1145  77355    6031         JSR     PUTC            OUTPUT QUESTION MARK
1146  77356    6001         JSR     CRLF
1147  77357    5612         JMP     RESET
1148                        *
1149                        *
1150  77360   34015 CRLF    LI      0,X'0D'
1151  77361    6025         JSR     PUTC            OUTPUT CR
1152  77362   34012         LI      0,X'0A'
1153  77363    6023         JSR     PUTC            OUTPUT LF
1154  77364  174120         RTS
1155                        *
1156                        *
1157  77365  172166 GETC    PUSH    SP,X1
1158  77366  136122         LDR     X1,MOSI
1159  77367  134123         LDR     0,MONE
1160  77370   14137         SKAZ    0,X'5F',X1
1161  77371    5401         JMP     $+2
1162  77372    5775         JMP     $-2
1163  77373  134120         LDR     0,MTWO
1164  77374   14137         SKAZ    0,X'5F',X1
1165  77375    5405         JMP     $+6
1166  77376   34177         LI      0,X'7F'
1167  77377  106137         AND     0,X'5F',X1
1168  77400  120137         ST      0,X'5F',X1
1169  77401  172206         POP     SP,X1
1170  77402  174120         RTS
1171  77403   34000         LI      0,0
1172  77404  120137         ST      0,X'5F',X1
1173  77405  172206         POP     SP,X1
1174  77406    5745         JMP     QUESTION
1175                        *
1176                        *
1177  77407  172166 PUTC    PUSH    SP,X1
1178  77410  170566         PUSH    SP,1
1179  77411  136100         LDR     X1,MOSO
1180  77412  134502         LDR     1,MFOR
1181  77413   14405         SKAZ    1,X'05',X1
1182  77414    5401         JMP     $+2
1183  77415    5775         JMP     $-2
1184  77416  120004         ST      0,X'04',X1
1185  77417  170606         POP     SP,1
1186  77420  172206         POP     SP,X1
1187  77421  174120         RTS
```

```
1189                        *
1190                        *       PNCHBK PUNCHES N INCHES OF NULL TAPE WHERE N = CONTENTS OF R2.
1191                        *       IF R2 IS NEGATIVE OR THE NUMBER OF INCHES IS NOT SPECIFIED, 6
1192                        *       INCHES WILL BE PUNCHED.  BEFORE STARTING THE PUNCH OPERATION
1193                        *       PNCHSTRT WAITS FOR ANY KEY TO BE HIT TO DIICATE THAT HE PUNCH
1194                        *       HAS BEEN TURNED ON.  ON COMPLETION OF THE PUNCH OPERATION PNCHS
1195                        *       WILL AGAIN WAIT FOR A KEY TO BE HIT INDICATING THAT THE PUNCH
1196                        *       HAS BEEN TURNED OFF.  THE OPERATION MAY BE ABORTED AT ANY TIME
1197                        *       BY HITTING ANY KEY.
1198                        *
1199  77422    6335  PNCHBK  JSR     CRLF
1200  77423  164007          LD      2,CONVERT,X2    CHECK FOR HEX ENTRY
1201  77424   70001          CPI     0,1
1202  77425    5002          BNZ     DEFAULT
1203  77426   71000          CPI     2,0             MAKE SURE NOT NEG
1204  77427    6401          BP      INCH
1205  77430   35010  DEFAULT LI      2,8
1206  77431  175222  INCH    SAL     2,3             CONVERT TO INCHES
1207  77432  171142          RNEG    2,2
1208  77433   34000          LI      0,0
1209  77434    6023          JSR     KBWAIT          WAIT FOR KEY TO BE HIT
1210  77435    6351  NXTBLNK JSR     PUTC            PUNCH A NULL
1211  77436  172166          PUSH    SP,X1
1212  77437  136051          LDR     X1,MDSI
1213  77440  134052          LDR     0,MONE
1214  77441   14137          SKAZ    0,X'5F',X1
1215  77442    5401          JMP     $+2
1216  77443    5402          JMP     $+3
1217  77444  172206          POP     SP,X1
1218  77445    5403          JMP     ABORT
1219  77446  172206          POP     SP,X1
1220  77447   34000          LI      0,0
1221  77450   75364          BINC    2,NXTBLNK


1223  77451  172166  ABORT   PUSH    SP,X1
1224  77452  136036          LDR     X1,MDSI
1225  77453   34000          LI      0,0
1226  77454  120137          ST      0,X'5F',X1
1227  77455  172206          POP     SP,X1
1228  77456    6001          JSR     KBWAIT          WAIT FOR KEY
1229  77457    1504          JMP     *RESET3


1231                        *
1232                        *
1233  77460  172166  KBWAIT  PUSH    SP,X1
1234  77461  136027          LDR     X1,MDSI
1235  77462  134030          LDR     0,NONE
1236  77463   14137          SKAZ    0,X'5F',X1
1237  77464    5401          JMP     $+2
1238  77465    5775          JMP     $-2
1239  77466   34000          LI      0,0
1240  77467  120137          ST      0,X'5F',X1
1241  77470  172206          POP     SP,X1
1242  77471  174120          RTS
```

```
1245                            *
1246                            *
1247                            *
1248    77472   165443  PNCHBC  LD      3,PNCHWCNT,X2
1249    77473    31775          ADI     3,-3
1250    77474   135130  PNCHBLK LDR     2,XFF
1251    77475   164443  NXTPNCH LD      1,PNCHWCNT,X2
1252    77476   170001          RMOV    0,1
1253    77477   170062          RAND    0,2
1254    77500     6306          JSR     PUTC
1255    77501   170001          RMOV    0,1
1256    77502   174247          ROT     0,8
1257    77503   170062          RAND    0,8
1258    77504     6302          JSR     PUTC
1259    77505    32401          ADI     X2,1
1260    77506    75766          BINC    3,$XTPNCH
1261    77507   136512          LDR     X2,RAMPT3
1262    77510   174120          RTS

1264    77511   117400  MOSI    DATA    X'0F00'
1265    77512   177760  MOSO    DATA    X'FFF0'
1266    77513      400  MONE    DATA    X'0100'
1267    77514     1000  MTWO    DATA    X'0200'
1268    77515        4  MFOR    DATA    X'0004'
```

```
1270                    *
1271                    *       PNCHMEM PUNCHES ON TAPE THE DATA IN THE MEMORY AREA BOUNDED BY
1272                    *       THE ADDRESSES IN 'FIRST' AND REG 2.  THE TAPE FORMAT IS COMPATI
1273                    *       WITH THE LOADER (LSH OF -WC / MSH OF -WC / LSH OF ADDRESS /
1274                    *       MSH OF ADDRESS / LSH OF CHECKSUM / MSH OF CHECKSUM / DATA...)
1275                    *       BEFORE STARTING THE PUNCH OPERATION PNCHMEM WAITS FOR A (ANY) K
1276                    *       TO BE HIT TO INDICATE THAT THE PUNCH HAS BEEN TURNED ON.
1277                    *       ON COMPLETION OF THE PUNCH OPERATION PNCHMEM WILL AGAIN WAIT FO
1278                    *       A KEY TO BE HIT INDICATING THAT THE PUNCH HAS BEEN TURNED OFF.
1279                    *       THE PUNCH OPERATION MAY BE ABORTED AT ANY TIME BY HITTING ANY K
1280                    *
1281  77516  161405  PNCHMEM  ST      2,LAST,X2       SAVE ADDR OF LAST LOC TO BE PUNCHED
1282  77517    6240           JSR     CRLF
1283  77520    6337           JSR     KBWAIT          WAIT FOR PUNCH TO BE TURNED ON
1284  77521  166003           LD      X1,FIRST,X2     X1 WILL = ADDR OF WORD TO BE PUNCHED
1285  77522  162044  NXTBLOC  ST      X1,PNCHADDR,X2
1286  77523  171404           RMOV    3,X1            R3 WILL = CHECKSUM
1287  77524   35017           LI      2,15
1288  77525  171024           RADD    2,X1            R2 WILL = ADDR OF LAST WD IN CUR PUNCH
1289  77526  155005           CMPR    2,LAST,X2       SEE IF < 16 WDS LEFT TO PUNCH
1290  77527    6445           BP      TOOMUCH         YES
1291  77530   34360           LI      0,-16
1292  77531  160043  INSRTWC  ST      0,PNCHWCNT,X2   STORE WORD COUNT IN PUNCH BUFFER
1293  77532  171420           RADD    3,0             UPDATE CHECKSUM
1294  77533  172166           PUSH    SP,X1
1295  77534  172166           PUSH    SP,0
1296  77535  134455           LDR     0,NONE
1297  77536  136352           LDR     X1,MOSI
1298  77537   14137           SKAZ    0,X'5FF,X1
1299  77540    5401           JMP     $+2
1300  77541    5405           JMP     $+6
1301  77542   34000           LI      0,0
1302  77543  120137           ST      0,X'5FF,X1
1303  77544  172206           POP     SP,0
1304  77545  172206           POP     SP,X1
1305  77546    5702           JMP     ABORT
1306  77547  172206           POP     SP,0
1307  77550  172206           POP     SP,X1

1309  77551  124000  NXTWORD  LD      0,0,X1
1310  77552  160046           ST      0,PNCHDATA,X2   STORE DATA WD IN PUNCH BUFFER
1311  77553  171420           RADD    3,0             UPDATE CHECKSUM
1312  77554   32401           ADI     X2,1            INCREMENT DESTINATION PTR
1313  77555   32401           ADI     X1,1            INCREMENT SOURCE PTR
1314  77556  171124           RCMP    2,X1            SEE IF FINISHED W/ CURRENT BLOCK
1315  77557    6771           BP      NXTWORD         NO

1317  77560  136441           LDR     X2,RAMPT3
1318  77561   34400           LI      1,0
1319  77562  170443           RSUB    1,3             COMPUTE CHECKSUM
1320  77563  160445           ST      1,PNCHKSUM,X2
1321  77564    6305           JSR     PNCHBC          PUNCH OUT THE BUFFER
1322  77565   34776           LI      1,-2            OUTPUT
1323  77566   34000           LI      0,0             BLOCK
```

```
1325   77567    6217   SPACER   JSR      PUTC          *SPACER
1326   77570   74776            HINC     1,SPACER
1327   77571  164435            LD       1,LAST,X2
1328   77572  170444            RSUB     1,X1          SEE IF FINISHED
1329   77573    6726            BP       NXTBLOC       NO
1330   77574    6263            JSR      KBWAIT        WAIT FOR PUNCH TO BE TURNED OFF
1331   77575    1426            JMP      *RESETS

1333   77576  165005   TOOMUCH  LD       2,LAST,X2
1334   77577  170004            RMOV     0,X1          COMPUTE NEGATIVE
1335   77600  170042            RSUB     0,2           OF WORD
1336   77601   30377            ADI      0,-1          COUNT
1337   77602    5726            JMP      INSRTWC
```

```
1339                        *
1340                        *       PNCHSTRT PUNCHES A START/STOP BLOCK IN LOADER COMPATIBLE FORMAT
1341                        *       IT WAITS FOR THE PUNCH TO BE TURNED ON AND OFF IN THE SAME MANN
1342                        *       AS FOR PNCHMEM AND PNCHBK
1343                        *
1344  77603     2022  PNCHSTRT JSR     *CRLF3
1345  77604     6253          JSR     KBWAIT          WAIT FOR PUNCH TO BE TURNED ON
1346  77605    34001          LI      0,1
1347  77606   160043          ST      0,PNCHRCNT,X2   SET UP 'START BLOCK' INDICATOR
1348  77607   164007          LD      0,CONVERT,X2
1349  77610     5001          BNZ     $+2             CHECK FOR START OR STOP BLOCK
1350  77611    35377          LI      2,-1            STOP BLOCK
1351  77612   161044          ST      2,PNCHADDR,X2
1352  77613    31001          ADI     2,1             CHECKSUM = -(REG2 +1)
1353  77614   171142          RNEG    2,2
1354  77615   161045          ST      2,PNCHKSUM,X2
1355  77616    35775          LI      3,-3
1356  77617     6254          JSR     PNCHBLK
1357  77620     6257          JSR     KBWAIT          WAIT FOR PUNCH TO BE TURNED OFF
1358  77621     1402          JMP     *RESET3

1360  77622     1640  RAMPT3  DATA    VARADDR
1361  77623    76120  INPRP3  DATA    INTERPRT
1362  77624    77172  RESET3  DATA    RESET
1363  77625      377  XFF     DATA    X'FF'
1364  77626    77360  CRLF3   DATA    CRLF
1365                        $END
1366                         END
                                              **END OF PASS TWO**

  1 ERRORS

    990 (  1736 OCT) WORDS OF CODE GENERATED
```

```
                **MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION**

                                **END OF PASS ONE**

NO ERRORS

**SYMBOL VALUES:

  4 H00    /    454  1        V00    /    455  1        P1     /    103  1        Q14    /   1045  1
  4 P2     /    110  1        Q24    /   1052  1        LADR   /     20  1        TABL   /   1101  1
  4 TABLA  /   1102  1        FCOUNT /     54  1        QCOS   /     43  1        SCMSK  /     56  1
  4 NEXT   /    114  1        RETURN /    117  1        VECDAT /     37  1        VECTR  /   1000  1
  4 VECTRA /     51  1        QGEN   /    400  1        QGENA  /     53  1        QH0    /     40  1
  4 QH00   /     45  1        PHASE  /     61  1        THETA  /     50  1        LIST   /     57  1
  4 LISTSA /     23  1        ALIST  /    456  1        ALISTA /     22  1        QLIST1 /    516  1
  4 QLIST2 /    560  1        QLISTA /     20  1        QLISTb /     63  1        QLINE  /     44  1
  4 DMACNT /    453  1        PMSK   /     62  1        QMSK   /     55  1        RMSK   /     64  1
  4 ANGL   /   1035  1        ANGLA  /     52  1        QNEG   /   1067  1        END    /    120  1
  4 INT60  /     65  1        SNEG   /   1071  1        ROOT   /    121  1        CONT   /   1074  1
  4 LOAD   /   1060  1        LOOP   /   1013  1        QPMSK  /     47  1        QSIN   /     42  1
  3 STACK  /    127  1        OV0    /     41  1        OV00   /     46  1
FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED
```

```
                **MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04/24/77 VERSION**

    1                       *       PROG DISPLY GEN TEST
    2                       *
    3                       *
    4                       *BASE PAGE DATA
    5           1                   $LOC    1
    6         1     65                DATA    INT60           60HZ INTERRUPT VECTOR
    7             5                 $LOC    5
    8         5     0                 DATA    0
    9            20                 $LOC    X'10'
   10        20     516               DATA    DLIST1          DISPLAY LIST ADDR
   11        21      22               DATA    16              WORD COUNT
   12        22     456               DATA    ALIST           ANG LIST ADDR
   13        23      57               DATA    LIST            OUTPUT LISTS ADDR
   14        24      23               DATA    LISTSA          OUTPUT LISTS ADDR
   15           40                 $LOC    X'20'
   16        40   20000               DATA    X'2000'         HV OP CODE
   17        41   24000               DATA    X'2600'         VV OP CODE
   18        42   50000               DATA    X'5000'         SIN OP CODE
   19        43   54000               DATA    X'5400'         COS OP CODE
   20        44  100000               DATA    X'8000'         LINE OP CODE
   21        45   10000               DATA    X'1000'         HOV OP CODE
   22        46   14000               DATA    X'1800'         VOV OP CODE
   23        47    3777               DATA    X'07FF'         OP CODE MASK
   24        50       0               DATA    0               ROTATION BIAS
   25        51    1000               DATA    VECTR           VECTOR SUBR ENTRY
   26        52    1035               DATA    ANGL            ANGLE SUBR ENTRY
   27        53     400               DATA    DGEN            DISPLAY GEN ENTRY
   28        54       3               DATA    3               FIELD COUNT
   29        55      20               DATA    X'10'           QUADRANT MASK
   30        56      17               DATA    X'F'            SINCOS MASK
   31        57     516               DATA    DLIST1          DISPLAY LIST 1
   32        60     560               DATA    DLIST2          DISPLAY LIST 2
   33        61       0               DATA    0               PHASE COUNT
   34        62       1               DATA    1               PHASE MASK
   35        63     560               DATA    DLIST2          DISPLAY GEN LIST ADDR
   36        64     377               DATA    X'FF'           ROTATION MASK
   37                        *
   38                        *60HZ INTERRUPT HANDLER
   39        65   64054      INT60    LD      A0,FCOUNT       GET FIELD COUNT
   40        66   30001               ADI     A0,1            INCREMENT FIELD COUNT
   41        67   60054               ST      A0,FCOUNT       SAVE
   42        70   72004               CPI     A0,4            15HZ POINT?
   43        71    5025               BNZ     RETURN
   44        72   54000               LI      A0,0
   45        73   60054               ST      A0,FCOUNT       RESET FIELD COUNT
   46        74   64061               LD      A0,PHASE        GET DISPLAY PHASE
   47        75   30001               ADI     A0,1            INCREMENT PHASE
   48        76   46062               AND     A0,PMSK         NORMALIZE
   49        77   60061               ST      A0,PHASE        SAVE
   50       100   65023               LD      X1,LISTSA       GET POINTER TO OUTPUT LISTS
   51       101   70001               CPI     A0,1            CHECK PHASE
   52       102    4005               BZ      P2              BRANCH IF PHASE 2
   53       103  124400      P1       LD      A1,0,X1
   54       104  125001               LD      A2,1,X1
   55       105   60420               ST      A1,DLISTA
   56       106   61063               ST      A2,DLIST8
   57       107    5404               JMP     NEXT
   58       110  124401      P2       LD      A1,1,X1
   59       111  125000               LD      A2,0,X1
   60       112   60420               ST      A1,DLISTA
   61       113   61063               ST      A2,DLIST8
```

```
 62   114    4020  NEXT    BLOC    DLISTA          START DMA
 63   115      55          JSRZ    *DGENA           UPDATE DISPLAY LIST
 64   116    5401          JMP     END
 65   117    4020  RETURN  BLOC    DLISTA          START DMA
 66   120    7400  END     RTI
 67                *
 68                *BOOTSTRAP
 69   121  137005  BOOT    LDR     SP,STACK        SET STACK POINTER
 70   122   34000          LI      A0,0
 71   123  174320          POUT    A0,0            ARM INTERRUPTS
 72   124    3410          IMSK    8               ENABLE INTERRUPTS
 73   125    5400          NOP
 74   126    5776          JMP     $-1             WAIT FOR 60HX INTR
 75   127   52000  STACK   DATA    X'5000'
 76                *
 77                *DISPLAY GEN ROUTINE
 78                *GENERATES RECTANGLE THAT ROTATES
 79                *BASE PAGE ASSIGNMENTS
 80          37    VECDAT  EQU     X'1F'
 81          40    DH0     EQU     VECDAT+1
 82          41    DV0     EQU     VECDAT+2
 83          42    DSIN    EQU     VECDAT+3
 84          43    DCOS    EQU     VECDAT+4
 85          44    DLINE   EQU     VECDAT+5
 86          45    DH00    EQU     VECDAT+6
 87          46    DV00    EQU     VECDAT+7
 88          47    DPMSK   EQU     VECDAT+8
 89          50    THETA   EQU     VECDAT+9
 90          51    VECTRA  EQU     VECDAT+10
 91          52    ANGLA   EQU     VECDAT+11
 92          53    DGENA   EQU     VECDAT+12
 93          54    FCOUNT  EQU     VECDAT+13
 94          55    QMSK    EQU     VECDAT+14
 95          56    SCMSK   EQU     VECDAT+15
 96          57    LIST    EQU     VECDAT+16
 97          61    PHASE   EQU     VECDAT+18
 98          62    PMSK    EQU     VECDAT+19
 99          63    DLISTB  EQU     VECDAT+20
100          64    RMSK    EQU     VECDAT+21
101          20    LADR    EQU     X'10'
102          20    DLISTA  EQU     LADR
103          22    ALISTA  EQU     LADR+2
104          23    LISTSA  EQU     LADR+3
105     400        $LOC    X'100'
106  400   66463  DGEN    LD      X2,DLISTB       SET UP DISPLAY LIST POINTER
107  401   32402          ADI     X2,2
108  402  134051          LDR     A0,H00          GET H00 OFFSET
109  403   46047          AND     A0,DPMSK
110  404   44045          OR      A0,DH00
111  405  170165          PUSH    X2,A0           OUTPUT TO DISPLAY LIST
112                *
113  406  134046          LDR     A0,V00          GET V00 OFFSET
114  407   46047          AND     A0,DPMSK
115  410   44046          OR      A0,DV00
116  411  170165          PUSH    X2,A0           OUTPUT TO DISPLAY LIST
117                *
118  412   66022          LD      X1,ALISTA       SET UP ARG LIST POINTER
119  413   34004          LI      A0,4            NO. OF VECTORS TO DRAW
120  414  120000          ST      A0,0,X1
121  415   34360          LI      A0,-16          INIT H0
122  416  120001          ST      A0,1,X1
123  417   34560          LI      A0,-16          INIT V0
124  420  120002          ST      A0,2,X1
125  421   34000          LI      A0,0            SET UP VECTOR 1
126  422   34440          LI      A1,32
127  423  120003          ST      A0,3,X1
128  424  120004          ST      A1,4,X1
129  425   34100          LI      A0,64           SET UP VECTOR 2
130  426   34440          LI      A1,32
```

```
 131     427   120005            ST      A0,5,X1
 132     430   120406            ST      A1,6,X1
 133     431    34177            LI      A0,127          SET UP VECTOR 3
 134     432    30001            ADI     A0,1
 135     433    34440            LI      A1,32
 136     434   120007            ST      A0,7,X1
 137     435   120410            ST      A1,8,X1
 138     436    34144            LI      A0,100          SET UP VECTOR 4
 139     437    30134            ADI     A0,92
 140     440    34440            LI      A1,32
 141     441   120011            ST      A0,9,X1
 142     442   120412            ST      A1,10,X1
 143                     *
 144     443       51            JSRZ    *VECTRA         BUILD SYMBOL DISPLAY LIST
 145     444   134006            LDR     A0,DMACNT
 146     445    60021            ST      A0,DLISTA+1     SET WORD COUNT
 147     446    65450            LD      A3,THETA        GET ROTATION BIAS
 148     447    31401            ADI     A3,1            CHANGE ROTATION BIAS
 149     450    47464            AND     A3,RMSK
 150     451    61450            ST      A3,THETA
 151     452   174120            RTS
 152                     *DATA STORAGE
 153     453       13    DMACNT  DATA    11
 154     454      777    H00     DATA    511
 155     455      777    V00     DATA    511
 156       516           ALIST   RES     32
 157     516        0    DLIST1  DATA    0
 158     517     4000            DATA    X'0800'
 159       560                   RES     32
 160     560        0    DLIST2  DATA    0
 161     561     4000            DATA    X'0800'
 162       622                   RES     32
 163                     *
 164                     *VECTOR SUBROUTINE
 165                     *BUILDS DISPLAY LIST FOR VECTOR CHAINS
 166                     *SUBR CALL ARGS:
 167                     * X1=ADDR OF INPUT PARAMETERS
 168                     * (X1)=NO. OF VECTORS
 169                     * (X1+1)=X POSITION
 170                     * (X1+2)=Y POSITION
 171                     * (X1+3)=VECTOR 1 ANGLE
 172                     * (X1+4)=VECTOR 1 LENGTH
 173                     * ----------------------
 174                     * ----------------------
 175                     * (X1+N+2)=VECTOR N ANGLE
 176                     * (X1+N+3)=VECTOR N LENGTH
 177                     *
 178                     * X2=ADDR OF OUTPUT DISPLAY LIST
 179                     *
 180      1000                   SLOC    X'200'
 181     1000   124000  VECTR    LD      A0,0,X1         GET NO. OF VECTORS
 182     1001   171140            RNEG    A2,A0           SET LOOP COUNT
 183                     *
 184     1002   124001            LD      A0,1,X1         GET X POSITION
 185     1003    46047            AND     A0,OPMSK        SET OP CODE FIELD
 186     1004    44040            OR      A0,OH0
 187     1005   170165            PUSH    X2,A0           OUTPUT TO DISPLAY LIST
 188                     *
 189     1006   124002            LD      A0,2,X1         GET Y POSITION
 190     1007    46047            AND     A0,OPMSK        SET OP CODE FIELD
 191     1010    44041            OR      A0,OV0
 192     1011   170165            PUSH    X2,A0           OUTPUT TO DISPLAY LIST
 193     1012    32003            ADI     X1,3
 194                     *
```

```
195   1013  124000  LOOP    LD      A0,0,X1         GET VECTOR DIRECTION
196   1014   42050          ADD     A0,THETA        ADD ROTATION BIAS
197   1015      52          JSRZ    *ANGLA           GET SIN,COS VALUES
198   1016  170245          SHRA    A0,6            RESCALE SIN VALUE
199   1017   46047          AND     A0,OPMSK        SET OP CODE FIELD
200   1020   44042          OR      A0,OSIN
201   1021  170165          PUSH    X2,A0           OUTPUT TO DISPLAY LIST
202   1022  170645          SHRA    A1,6            RESCALE COS VALUE
203   1023   46047          AND     A1,OPMSK        SET OP CODE FIELD
204   1024   44043          OR      A1,OCOS
205   1025  170565          PUSH    X2,A1           OUTPUT TO DISPLAY LIST
206                 *
207   1026  124001          LD      A0,1,X1         GET VECTOR LENGTH
208   1027   46047          AND     A0,OPMSK        SET OP CODE FIELD
209   1030   44044          OR      A0,OLINE
210   1031  170165          PUSH    X2,A0           OUTPUT TO DISPLAY LIST
211   1032   32002          ADI     X1,2
212   1033   75357          BINC    A2,LOOP         LOOP BACK IF MORE VECTORS
213   1034  174120          RTS
214                 *
215                 *ANGLE SUBROUTINE
216                 *DECODES DIRECTION INTO SIN(A0),COS(A1)
217   1035  172166  ANGL    PUSH    SP,X1           SAVE X1 REG
218   1036  171166          PUSH    SP,A2
219   1037  171566          PUSH    SP,A3
220   1040  136037          LDR     X1,TABLA        GET SIN,COS TABLE POINTER
221   1041  170221          SHR     A0,2            RESCALE ARG
222   1042  171000          RMOV    A2,A0           SAVE COPY OF ARG
223   1043   46055          AND     A0,QMSK         CHECK FOR QUADRANT
224   1044    5005          BNZ     Q24             BRANCH IF IIN QUAD 2 OR 4
225   1045  170002  Q13     RMOV    A0,A2
226   1046   46056          AND     A0,SCMSK
227   1047  174220          SHL     A0,1            ADJUST INDEX
228   1050  172020          RADD    X1,A0           ADD INDEX TO TABLE POINTER
229   1051    5406          JMP     LOAD
230   1052  170002  Q24     RMOV    A0,A2
231   1053   46056          AND     A0,SCMSK
232   1054  174220          SHL     A0,1
233   1055  170140          RNEG    A0,A0           NEGATE INDEX
234   1056   32040          ADI     X1,32           MOVE POINTER TO END OF TABLE
235   1057  172020          RADD    X1,A0           ADD INDEX TO TABLE POINTER
236   1060  124000  LOAD    LD      A0,0,X1         PUT SIN VALUE IN A0
237   1061  124401          LD      A1,1,X1         PUT COS VALUE IN A1
238                 *CHECK FOR SIGNS
239   1062   71040          CPI     A2,32           QUADRANTS 3 OR 4?
240   1063    6405          BP      SNEG            BRANCH IF YES
241   1064   71020          CPI     A2,16           QUADRANT 2?
242   1065    6401          BP      CNEG            BRANCH IF YES
243   1066    5405          JMP     CONT            DONE
244   1067  170541  CNEG    RNEG    A1,A1           NEGATE COS
245   1070    5403          JMP     CONT            DONE
246   1071  170140  SNEG    RNEG    A0,A0           NEGATE SIN
247   1072   71060          CPI     A2,48           QUADRANT 3?
248   1073    7375          BN      CNEG            BRANCH IF YES
249   1074  171606  CONT    POP     SP,A3
250   1075  171206          POP     SP,A2
251   1076  172206          POP     SP,X1           RESTORE X1
252   1077  174120          RTS
253   1100    1101  TABLA   DATA    TABL
254   1101       0  TABL    DATA    0               SIN 0
255   1102   40000          DATA    16384           COS 0
```

```
256   1103   3105          DATA    1605        SIN 5.625
257   1104   57661         DATA    16505       COS 5.625
258   1105   6174          DATA    3196        SIN 11.25
259   1106   57573         DATA    15173       COS 11.25
260   1107   11224         DATA    4756        SIN 16.875
261   1110   56477         DATA    15679       COS 16.875
262   1111   14176         DATA    6270        SIN 22.5
263   1112   55441         DATA    15137       COS 22.5
264   1113   17053         DATA    7723        SIN 28.125
265   1114   54161         DATA    14449       COS 28.125
266   1115   21616         DATA    9102        SIN 33.75
267   1116   52467         DATA    13623       COS 33.75
268   1117   24252         DATA    10394       SIN 39.375
269   1120   50571         DATA    12665       COS 39.375
270   1121   26501         DATA    11585       SIN 45
271   1122   26501         DATA    11585       COS 45
272   1123   50571         DATA    12665       SIN 50.625
273   1124   24252         DATA    10394       COS 50.625
274   1125   52467         DATA    13623       SIN 56.25
275   1126   21616         DATA    9102        COS 56.25
276   1127   54161         DATA    14449       SIN 61.875
277   1130   17053         DATA    7723        COS 61.875
278   1131   55441         DATA    15137       SIN 67.5
279   1132   14176         DATA    6270        COS 67.5
280   1133   56476         DATA    15678       SIN 73.125
281   1134   11224         DATA    4756        COS 73.125
282   1135   57305         DATA    16069       SIN 78.75
283   1136   6174          DATA    3196        COS 78.75
284   1137   57661         DATA    16505       SIN 84.375
285   1140   3106          DATA    1606        COS 84.375
286   1141   40000         DATA    16384       SIN 90
287   1142   0             DATA    0           COS 90
288                        $END
289                        END
```

END OF PASS TWO

NO ERRORS

$08 (464 OCT) WORDS OF CODE GENERATED

APPENDIX B

COMPILED PDP-11/45 LISTING OF CROSS ASSEMBLER

Following is a listing of the MDSC Cross Assembler program coded in FORTRAN as compiled by the PDP-11/45 computer at WPAFB with the RSX-11M operating system with the FORTRAN IV-PLUS compiler.

AD-A051 886 DAYTON UNIV OHIO RESEARCH INST F/G 19/5
FIRE CONTROL SYSTEM ANALYSIS VOLUME II. COMPUTER PROGRAMMING TA--ETC(U)
NOV 77 C KING F33615-77-C-1056
UNCLASSIFIED AFAL-TR-78-16-VOL-2 NL

2 OF 2
AD
A051 886

END
DATE
FILMED
5-78
DDC

```
FORTRAN IV-PLUS V02-04          10:47:25    21-JUL-77          PAGE 1
CARDS.FTN       /TR:BLOCKS/WR

0001            PROGRAM ASSEM
      C*****************************************************************************
      C
      C
      C       HUGHES AIRCRFT COMPANY-RADAR DIVISION-DISPLAY SYSTEMS LABORATORY
      C
      C
      C*****************************************************************************
      C
      C
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,CHAR
0004            LOGICAL STOFLG,FLG,CODFLG,LISFLG,TABFLG,ASMFLG,EXFLG
0005            DIMENSION OPTBLA(53), OPTBLB(53), SYMTAD(508),LISCNG(10)
0006            DOUBLE PRECISION SYMTAB,SYMTBB,OPRND,LOCCTR,TEMPA,TEMPB
0007            DOUBLE PRECISION OPCODE,ORG,EQU,DATA,RES,PAGE,END,OPTN,BLNK,BKP,
               *LOC,OPTBLA,XDATA,TITL,DOLND
0008            COMMON/A/READIN(20),I,NREAD,NWRITE
0009            COMMON/B/LN,CNT,PASMDE
0010            COMMON/C/CODE,WDCNT
0011            COMMON/D/SYMTAB(508),SYMTBB(508),SYMTBC(508),ENDTB,TABLND
0012            COMMON/E/LOCCTR
      C INITIALIZE OPCODE LOOKUP TABLE WITH SYMBOLIC CODES
0013            DATA OPTBLA/
               *5HADD  ,5HSUB  ,5HOR   ,5HAND  ,5HCMPR ,5HMULT ,5HDADD ,5HDIV  ,
               *5HBKAZ ,
               *5HLD   ,5HLDR  ,5HST   ,
               *5HRADD ,5HRSUB ,5HRNEG ,5HRABS ,5HRCMP ,5HRAND ,5HROR  ,5HRNOT ,
               *5HRMOV ,5HPUSH ,5HPOP  ,
               *5HSHR  ,5HSHL  ,5HSHRA ,5HROT  ,5HDSHR ,5HDSHL ,
               *5HLI   ,5HADI  ,5HCPI  ,
               *5HRINC ,5HLIM  ,
               *5HPIN  ,5HPOUT ,5HSIN  ,5HSOUT ,
               *5HBZ   ,5HBNZ  ,5HJMP  ,5HBP   ,5HBN   ,5HJSRZ ,5HJSR  ,
               *5HIOCP ,5HSKR  ,5HBLOC ,5HLOCK ,
               *5HIMSK ,
               *5HRTS  ,5HRTI  ,5HNOP
               */
      C
      C INITIALIZE OPCODE LOOKUP TABLE WITH MACHINE INSTRUCTION SKELETONS
0014            DATA OPTBLB/
               * "2000,    "0,  "4000,  "6000, "14000, "12000, "10000, "16000,
               * "10000,
               * "24000, "24000, "20000,
               *"170020,"170040,"170140,"174140,"170120,"170060,"170100,"174100,
               *"170000,"170160,"170200,
               *"170220,"174220,"170240,"174240,"170260,"174260,
               * "34000, "30000, "70000,
               * "74000,"130000,
               *"170300,"170320,"174340,"170340,
```

```
     *   "400,  "1000,  "1400,  "2400,  "3000,     "0,  "2000,
     *"170360,"174360,  "4000,  "3400,
     *   "3400,
     *"174120,  "7400,  "5400
     */
      C
      C DEFINE ASSEMBLER DIRECTIVES
0015       DATA ORG ,EQU ,DATA,RES ,PAGE,END ,OPTN,BLNK,SKP ,LOC ,XDATA,TITL,
     *DOLND
     */5HORG  ,5HEQU  ,5HDATA ,5HRES  ,5HPAGE ,5HEND  ,5HOPTN ,5H    ,
     *5HSKP  ,5HSLOC ,5HXDATA,5HTITL ,5HSEND                          /
      C
      C INITIALIZE SYMBOL TABLE WITH PREDEFINED REGISTER DESIGNATOR SYMBOLS
0016       SYMTAB(501)=2HA0
0017       SYMTAB(502)=2HA1
0018       SYMTAB(503)=2HA2
0019       SYMTAB(504)=2HA3
0020       SYMTAB(505)=2HX1
0021       SYMTAB(506)=2HX2
0022       SYMTAB(507)=2HSP
0023       SYMTAB(508)=2HPC
0024       SYMTBB(501)=0
0025       SYMTBB(502)=1
0026       SYMTBB(503)=2
0027       SYMTBB(504)=3
0028       SYMTBB(505)=4
0029       SYMTBB(506)=5
0030       SYMTBB(507)=6
0031       SYMTBB(508)=7
0032       SYMTBC(501)=1
0033       SYMTBC(502)=1
0034       SYMTBC(503)=1
0035       SYMTBC(504)=1
0036       SYMTBC(505)=1
0037       SYMTBC(506)=1
0038       SYMTBC(507)=1
0039       SYMTBC(508)=1
      C
      C LIMIT OF SYMBOL TABLE WORKING AREA
0040       TABLND=500
0041       NREAD=5
0042       NWRITE=6
      C
      C
      C PASS ONE SETUP
0043  1    CONTINUE
      C READ A LINE, AND STOP IF EOF
0044       READ(NREAD,20,END=3) READIN
0045       GO TO 4
0046  3    STOP
```

```
      C CLEAR SYMBOL TABLE TYPE FLAGS COLUMN
0047  4     DO 5 D=1,TABLND
0048        SYMTBC(D)=0
0049  5     CONTINUE
      C INITIALIZE ASSEMBLER OPTION FLAGS
0050        CODELG=.TRUE.
0051        LISFLG=.TRUE.
0052        TABFLG=.TRUE.
0053        ASMFLG=.FALSE.
0054        EXFLG=.FALSE.
      C INITIALIZE COUNTERS
0055        LOCCTR=0
0056        LN=0
0057        CNT=0
0058        P=1
0059        PASMDE=1
      C
0060        WRITE(NWRITE,7)
0061  7     FORMAT(32X,68H**MMP CROSS-ASSEMBLER (PDP FORTRAN IV PLUS-BASED) 04
           */24/77 VERSION**/)
0062        GOTO 22
      C
      C
      C PASS ONE MAIN LOOP
      C
      C READ A LINE OF SOURCE CODE
0063  10    READ(NREAD,20,END=135)READIN
      C CLEAR ERROR QUEUE AND PRINT PENDING MESSAGES
0064        CALL ERRCLR
0065  20    FORMAT(20A4)
      C
      C STORE SOURCE CARD IMAGES FOR PASS TWO
0066  22    WRITE(1) READIN
      C UPDATE LOCATION COUNTER AND LINE NUMBER , RESET SCANNING POINTER
0067        LN=LN+1
0068        I=0
0069        ENDTB=P
0070        STOFLG=.FALSE.
0071        CALL FETCH(I,1,CHAR)
      C TEST FOR COMMENT LINE
0072        IF(CHAR.EQ.1H*) GOTO 10
      C TEST FOR PRESENCE OF LABEL
0073        IF(CHAR.EQ.1H ) GOTO 30
      C STORE LABEL IN SYMBOL TABLE
0074        CALL GETFLD(SYMTAB(P))
      C STORE LINE NUMBER OF SYMBOL DEFINITION FOR CONCORDANCE
0075        SYMTAD(P)=LN
0076        STORLG=.TRUE.
      C
      C CHECK FOR SYMBOL TABLE OVERFLOW
```

```
0077            IF(P.LT.(TABLND)) GOTO 30
0078            CALL ERROR(11)
0079            CALL ERRCLR
0080            WRITE(NWRITE,25)TABLND
0081      25    FORMAT(2X,30HCURRENT SYMBOL TABLE CAPACITY=,I4,7HSYMBOLS)
0082            STOP
          C
          C GET OPCODE AND SESRCH DIRECTIVE LOOKUP TABLE
0083      30    CALL GETFLD(OPCODE)
0084            IF(OPCODE.EQ.ORG   ) GOTO 40
0085            IF(OPCODE.EQ.LOC   ) GOTO 40
0086            IF(OPCODE.EQ.EQU   ) GOTO 70
0087            IF(OPCODE.EQ.XDATA ) GOTO 103
0088            IF(OPCODE.EQ.DATA  ) GOTO 105
0089            IF(OPCODE.EQ.RES   ) GOTO 110
0090            IF(OPCODE.EQ.PAGE  ) GOTO 10
0091            IF(OPCODE.EQ.SKP   ) GOTO 10
0092            IF(OPCODE.EQ.END   ) GOTO 140
0093            IF(OPCODE.EQ.DOLND ) GOTO 10
0094            IF(OPCODE.EQ.OPTN  ) GOTO 132
0095            IF(OPCODE.EQ.TITL  ) GOTO 10
0096            IF(OPCODE.NE.BLNK  ) GOTO 35
          C IF NO OPCODE-ERROR
0097            CALL ERROR(8)
          C
          C IF LABEL WAS PRESENT FOR CURRENT STATEMENT,STORE ITS VALUE
0098      35    IF(STOFLG) CALL STORE(LOCCTR,1,P)
          C
          C ASSUME A MACHINE INSTRUCTION ON THIS LINE, AND INCREMENT LOCATION COUN
0099            LOCCTR=LOCCTR+1
0100            GOTO 10
          C
          C PROCESS ORIGIN STATEMENTS
          C INTERPRET OPERAND
0101      40    CALL INTERP(OPRND,TYPE)
0102            IF(TYPE.EQ.1) GOTO 45
          C IF OPERAND CONTAIND AN UNDEFINED TERM, SET VALUE TO ZERO-THAT'S AN ERR
0103            CALL ERROR(6)
0104            GOTO 50
0105      45    IF(OPRND.GE.0)GOTO 60
          C IF OPERAND MINUS, SET VALUE TO ZERO- THAT'S AN ERROR
0106            CALL ERROR(7)
0107      50    OPRND=0
          C UPDATE LOCATION LOUNTER
0108      60    LOCCTR=OPRND
          C IF LABEL WAS PRESENT, STORE ITS VALUE
0109            IF(STOFLG) CALL STORE(LOCCTR,1,P)
0110            GOTO 10
          C
          C PROCESS EQUIVALENCE STATEMENTS
```

```
0111     70     IF(STOFLG) GOTO 75
         C IF NO LABEL WAS PRESENT, IGNORE STATEMENT- THAT'S AN ERROR
0112            I=0
0113            CALL ERROR(9)
0114            GOTO 10
         C INTERPRET OPERAND
0115     75     CALL INTERP(OPRND,TYPE)
0116            IF(TYPE.NE.0) GOTO 90
         C IF IT CONTAINS UNDEFINED SYMBOLIC TERMS OR IS NEGATIVE, SET TO ZERO- E
0117            CALL ERROR(6)
0118            OPRND=0
0119            GOTO 100
0120     90     IF(OPRND.GE.0) GOTO 100
0121            CALL ERROR(2)
0122            OPRND=0
0123            TYPE=1
         C STORE VALUE OF LABEL
0124     100    CALL STORE(OPRND,TYPE,P)
0125            GOTO 10
         C
         C SET FLAG TO PROCESS EXTENDED DATA STATEMENTS
0126     103    EXFLG=.TRUE.
         C
         C PROCESS DATA STATEMENTS
         C
         C IF LABEL WAS PRESENT,STORE ITS VALUE
0127     105    IF(STOFLG) CALL STORE(LOCCTR,1,P)
0128     107    CALL TAB
0129            CALL GETLNG(J)
0130            I=I+J
0131            LOCCTR=LOCCTR+1
0132            IF((I.GE.78).OR.(.NOT.EXFLG)) GOTO 108
0133            GOTO 107
0134     108    EXFLG=.FALSE.
0135            GOTO 10
         C
         C PROCESS RESERVE STATEMENTS
         C INTERPRET OPERAND
0136     110    CALL INTERP(OPRND,TYPE)
         C IF LABEL WAS PRESENT, STORE ITS VALUE
0137            IF(STOFLG) CALL STORE(LOCCTR,1,P)
0138            IF(TYPE.EQ.1)GOTO 120
         C IF IT CONTAINS UNDEFINED TERMS, OR IS NEGATIVE, SET TO ZERO- ERROR
0139            CALL ERROR(6)
0140            OPRND=1
0141     120    IF(OPRND.GT.0)GOTO 130
0142            CALL ERROR(2)
0143            OPRND=1
         C UPDATE LOCATION COUNTER
0144     130    LOCCTR=LOCCTR+OPRND
```

```
0145            GOTO 10
                C
                C PROCESS OPTION STATEMENTS
0146    132     IF(ASMFLG) GOTO 135
0147            ASMFLG=.TRUE.
                C INHIBIT ALL OPTIONS
0148            CODFLG=.FALSE.
0149            LISFLG=.FALSE.
0150            TABFLG=.FALSE.
0151    133     CALL TAB
0152            IF(I.GE.78)GOTO 10
0153            CALL FETCH(I,1,OPRND)
0154            I=I+1
                C ALLOW SELECTED OPTIONS
0155            IF(OPRND.EQ.1HL) LISFLG=.TRUE.
0156            IF(OPRND.EQ.1HB) CODFLG=.TRUE.
0157            IF(OPRND.EQ.1HT) TABFLG=.TRUE.
0158            GOTO 133
                C
0159    135     CALL ERROR(12)
                C
                C PROCESS END STATEMENTS
                C CLEAR ERROR QUEUE-PRINT PENDING MESSAGES
0160    140     CALL ERRCLR
                C
0161            LASTLN=LN
0162            ENDTB=ENDTB-1
0163            WRITE(NWRITE,142)
0164    142     FORMAT(56X,19H**END OF PASS ONE**/)
0165            IF(CNT.EQ.0) WRITE(NWRITE,144)
0166    144     FORMAT(2X,9HNO ERRORS//)
0167            IF(CNT.NE.0) WRITE(NWRITE,146)CNT
0168    146     FORMAT(2X,I3,7H ERRORS//)
0169    148     FLG=.FALSE.
                C
                C SYMBOL TABLE CLEAN UP
                C
                C RESOLVE FORWARD REFERENCES
0170            DO 170 P=1,ENDTB
0171            IF(SYMTBC(P).NE.0) GOTO 170
0172            PNT=0
0173            PNT=PNT+1
0174            IF(PNT.EQ.ENDTB+1) PNT=TABLND+1
0175            IF(PNT.GT.TABLND+8)GOTO 170
0176            IF(SYMTAB(PNT).NE.SYMTBB(P)) GOTO 170
0177            SYMTBB(P)=SYMTBB(PNT)
0178            SYMTBC(P)=SYMTBC(PNT)
0179            FLG=.TRUE.
0180    170     CONTINUE
0181            IF(FLG) GOTO 148
```

```
        C
        C FLAG UNDEFINED AND DOUBLY DEFINED TERMS
0182            DO 180 P=1,ENDTB
0183            IF(SYMTBC(P).EQ.0) SYMTBB(P)=0
0184            P1=P+1
0185            DO 180 PNT=P1,ENDTB
0186            IF(SYMTAB(P).EQ.SYMTAB(PNT)) SYMTBC(P)=2
0187    180     CONTINUE
        C
        C SORT SYMBOL TABEL
        C
0188            IF(.NOT.(TABFLG)) GOTO 201
0189            IF(ENDTB.LT.2) GOTO 195
        C SORT SYMBOL TABLE BY NAMES
0190            ENDTB1=ENDTB-1
0191            DO 194 P=1,ENDTB1
0192            IF(SYMTAB(P+1).GT.SYMTAB(P)) GOTO 194
0193            TEMPA=SYMTAB (P+1)
0194            TEMPB=SYMTBB(P+1)
0195            TEMPC=SYMTBC(P+1)
0196            TEMPD=SYMTAD(P+1)
0197            DO 185 PTEMP=1,P
0198            PNT=P+1-PTEMP
0199            IF(TEMPA.GE.SYMTAB(PNT)) GOTO 193
0200            SYMTAB (PNT+1)=SYMTAB (PNT)
0201            SYMTBB(PNT+1)=SYMTBB(PNT)
0202            SYMTBC(PNT+1)=SYMTBC(PNT)
0203            SYMTAD(PNT+1)=SYMTAD(PNT)
0204    185     CONTINUE
0205            PNT=0
0206    193     SYMTAB (PNT+1)=TEMPA
0207            SYMTBB(PNT+1)=TEMPB
0208            SYMTBC(PNT+1)=TEMPC
0209            SYMTAD(PNT+1)=TEMPD
0210    194     CONTINUE
0211    195     IF(.NOT.TABFLG) GOTO 201
0212            WRITE(NWRITE,196)
        C
        C PRINT SYMBOL TABLE
0213    196     FORMAT(2X,16H**SYMBOL VALUES:/)
0214            KT=3
0215            DO 198 P=1,ENDTB,4
0216            IF(ENDTB-P.LT.4) KT=ENDTB-P
0217            KT1=KT+1
0218            WRITE(NWRITE,197)KT1,((SYMTAB(P+D-1),IDINT(SYMTBB(P+D-1)),SYMTBC
               *(P+D-1)),D=1,KT1)
0219    197     FORMAT(2X,I2,4(1H ,A8,1H/,O8,2X,I1,8X))
0220    198     CONTINUE
0221            WRITE(NWRITE,200)
0222    200     FORMAT(2X,52H FLAG CODE: 0-UNDEFINED, 1-DEFINED, 2-DOUBLY DEFINED)
```

```
        C
        C
        C PASS TWO SETUP
        C
        C
0223    201   CONTINUE
0224          WRITE(NWRITE,630)
0225          WRITE(NWRITE,7)
        C REWIND SOURCE CARD IMAGE SCRATCH FILE
0226          REWIND 1
0227          LN=0
0228          LOCCTR=0
0229          CNT=0
0230          WDCNT=0
0231          PASMDE=3
0232          IF(LISFLG) PASMDE=2
0233          PT=1
        C
        C PASS TWO MAIN LOOP
        C
        C READ SOURCE CARD IMAGE
0234    210   READ (1) READIN
        C
        C CLEAR ERROR QUEUE AND PRINT PENDING MESSAGES
0235          CALL ERRCLR
0236          I=0
0237          CODE=0
0238          LN=LN+1
        C CHECK FOR EOF ON SOURCE SCRATCH FILE, TERMINATE PASS TWO IF DETECTED
0239          IF(LN.LE.LASTLN) GOTO 215
0240          GOTO 805
0241    215   CALL FETCH(0,1,CHAR)
        C CHECK FOR COMMENT LINES
0242          IF(CHAR.EQ.1H*) GOTO 660
        C                              AND LABELS
0243          IF(CHAR.EQ.1H ) GOTO 220
        C
        C IGNORE LABELS
0244          CALL GETLNG(J)
0245          I=I+J
        C GET OPCODE FIELD
0246    220   CALL GETFLD(OPCODE)
0247          IF(OPCODE.NE.BLNK  ) GOTO 225
        C OPCODE MISSING
0248          GOTO 255
0249    225   P=0
        C
        C SEARCH OPCODE LOOKUP TABLE
0250    230   P=P+1
        C
```

```
0251            IF(P.GT.53) GO TO 250
0252            IF(OPCODE.NE.OPTBLA(P)) GOTO 230
0253            CODE=OPTBLB(P)
0254            IF(P.EQ.53) GO TO 700
0255            IF((P.LE.9).OR.(P.GT.38)) GOTO 240
        C  FIRST OPERAND IS ANY REGISTER
        C  GET RA REGISTER
0256            CALL TAB
0257            CALL FETCH(I,1,CHAR)
0258            IF(CHAR.EQ.1H*) CALL ERROR(13)
0259            CALL GETRN(REG)
0260            IF((P.EQ.22).OR.(P.EQ.23)) GOTO 315
0261            CALL MERGE(REG,8)
        C IF MEMORY REFERENCE GROUP
0262    240     IF(P.LE. 9) GOTO 260
        C IF LOAD/STORE GROUP
0263            IF(P.LE.12) GOTO 280
        C IF SHIFT/IMMEDIATE GROUP
0264            IF(P.LE.23) GOTO 310
        C IF REGISTER REFERENCE GROUP
0265            IF(P.LE.32) GOTO 320
        C IF BINC/LIM
0266            IF(P.LE.34) GOTO 340
        C IF I/O GROUP
0267            IF(P.LE.38) GOTO 350
        C IF BRANCH GROUP
0268            IF(P.LE.45) GOTO 390
        C IF I/O SPECIAL GROUP
0269            IF(P.LE.49) GOTO 410
        C IF IMSK
0270            IF(P.LE.50) GOTO 440
        C IF RETURN INSTRUCTION
0271            GOTO 700
        C
0272    250     IF(OPCODE.EQ.DATA  ) GOTO 470
0273            IF(OPCODE.EQ.XDATA ) GOTO 465
0274            IF(OPCODE.EQ.EQU   ) GOTO 500
0275            IF(OPCODE.EQ.RES   ) GOTO 520
0276            IF(OPCODE.EQ.SKP   ) GOTO 550
0277            IF(OPCODE.EQ.ORG   ) GOTO 590
0278            IF(OPCODE.EQ.LOC   ) GOTO 590
0279            IF(OPCODE.EQ.PAGE  ) GOTO 620
0280            IF(OPCODE.EQ.OPTN  ) GOTO 660
0281            IF(OPCODE.EQ.UOLND ) GOTO 660
0282            IF(OPCODE.EQ.END   ) GOTO 800
0283            IF(OPCODE.EQ.TITL  ) GOTO 210
0284    255     CALL ERROR(8)
0285            CODE=3840
0286            LOCCTR=LOCCTR+1
0287            GOTO 660
```

```
        C
        C PROCESS MEMORY REFERENCE GROUP
0288    260   CALL TAB
        C INTERPRET REGISTER DESIGNATOR
0289          CALL GETAN(REG)
0290          IF(((P.GE.6).AND.(P.LE.8)).AND.((REG*0.5).NE.(INT(REG*0.5))))
             *CALL ERROR(10)
0291          CALL MERGE(REG,8)
        C CHECK FOR INDIRECT FLAG
0292          CALL FETCH(I,1,CHAR)
0293          IF(CHAR.EQ.1H*) CALL ERROR(13)
        C
        C ASSEMBLE ADDRESS
0294          CALL SMBL
0295          CALL GETXN(REG)
        C ASSEMBLE REGISTER CODE
0296          N=14
0297          IF(P.EQ.9) N=10
0298          CALL MERGE(REG,N)
0299          GOTO 700
        C
        C PROCESS LOAD/STORE GROUP
0300    280   CALL TAB
        C CHECK FOR INDIRECT FLAG
0301          CALL FETCH(I,1,CHAR)
0302          IF(CHAR.EQ.1H*) I=I+1
0303          IF((P.EQ.11).OR.(CHAR.EQ.1H*)) GOTO 283
        C ASSEMBLE ADDRESS
0304          CALL SMBL
0305          GOTO 285
        C ASSEMBLE RELATIVE ADDRESS
0306    283   CALL RSMBL
0307    285   IF(CHAR.EQ.1H*) GOTO 300
0308          IF(P.EQ.11) GOTO 295
        C INTERPRET INDEX REGISTER DESIGNATOR
0309          CALL GETXN(REG)
0310          CALL MERGE(REG,14)
0311          GOTO 700
0312    295   CALL MERGE(9,12)
0313          GOTO 700
0314    300   CALL GETXN(REG)
0315          IF(REG.EQ.1) GOTO 305
0316          CALL ERROR(13)
0317          CALL MERGE(1,14)
0318    305   IF(P.NE.10) GOTO 700
0319          CALL ERROR(13)
0320          CALL MERGE(1,14)
0321          GOTO 700
        C
        C PROCESS REGISTER REFERENCE GROUP
```

```
        C INTERPRET REGISTER DESIGNATOR
0322    310   CALL GETRN(REG)
0323          CALL MERGE(REG,0)
0324          GOTO 700
        C PROCESS PUSH AND POP
        C
0325    315   CALL MERGE(REG,0)
0326          CALL GETRN(REG)
0327          CALL MERGE(REG,8)
0328          GOTO 700
        C
        C PROCESS SHIFT-ROTATE GROUP
0329    320   IF(((P.EQ.28).OR.(P.EQ.29)).AND.((REG*0.5).NE.(INT(REG*0.5))))
             *CALL ERROR(10)
        C INTERPRET OPERAND
0330          CALL INTERP(OPRND,TYPE)
0331          IF(TYPE.EQ.1) GOTO 322
0332          CALL ERROR(5)
0333          OPRND=0
0334    322   IF(P.GE.30) GOTO 330
0335          OPRND=OPRND-1
        C CHECK RANGE OF OPERAND
0336          IF((OPRND.GE.0).AND.(OPRND.LE.15)) GOTO 325
0337          CALL ERROR(2)
0338    324   OPRND=0
0339    325   CALL MERGE(MASKA(OPRND,255),0)
0340          GOTO 700
        C
        C PROCESS IMMEDIATE OPERANDS
0341    330   IF((OPRND.GE.-256).AND.(OPRND.LE.255)) GOTO 325
0342          CALL ERROR(2)
0343          GOTO 324
0344    340   CALL RSMBL
0345          GOTO 700
        C
        C PROCESS I/O GROUP
0346    350   CALL INTERP(OPRND,TYPE)
0347          IF(TYPE.NE.0) GOTO 352
0348          CALL ERROR(5)
0349          OPRND=0
        C CHECK RANGE OF OPERAND
0350    352   IF((OPRND.LT.31).AND.(OPRND.GE.0)) GOTO 360
0351    355   CALL ERROR(2)
0352          OPRND=0
0353    360   IF(OPRND.LE.15) GOTO 370
0354          IF(P.GE.37) GOTO 355
        C ASSEMBLE GROUP FLAG BIT
0355          CALL MERGE(1,11)
0356    370   CALL MERGE(MASKA(OPRND,15),0)
0357          GOTO 700
```

```
        C
        C PROCESS BRANCH GROUP
0358    390   CALL TAB
0359          CALL FETCH(I,1,CHAR)
0360          IF(CHAR.EQ.1H*) GOTO 400
0361          IF(P.NE.44) GOTO 395
        C JUMP SUBROUTINE INSTRUCTION MUST BE INDIRECT
0362          CALL ERROR(13)
        C ASSEMBLE ADDRESS
0363          CALL SMBL
0364          GOTO 700
0365    395   CALL MERGE(1,11)
0366          GOTO 405
0367    400   I=I+1
0368    405   IF(P.NE.44) CALL RSMBL
0369          IF(P.EQ.44) CALL SMBL
0370          GOTO 700
        C
        C PROCESS BINC AND LIM
0371    410   IF(P.LE.47) GOTO 420
        C EVALUATE OPERAND
0372          CALL SMBL
0373          GOTO 700
        C
        C PROCESS I/O SPECIAL GROUP
0374    420   CALL TAB
0375          CALL INTERP(OPRND,TYPE)
0376          IF(TYPE.NE.0) GOTO 430
0377          CALL ERROR(5)
0378          OPRND=0
0379          GOTO 435
        C CHECK I/O ADDRESS FOR RANGE
0380    430   IF((OPRND.GE.0).AND.(OPRND.LE.127)) GOTO 435
0381          CALL ERROR(2)
0382          GOTO 700
        C ASSEMBLE HIGH AND LOW ORDER FIELDS
0383    435   CALL MERGE(MASKA(OPRND,15),0)
0384          CALL MERGE(MASKA(OPRND,112),4)
0385          GOTO 700
        C
        C PROCESS IMSK
0386    440   CALL INTERP(OPRND,TYPE)
0387          IF(TYPE.NE.0) GOTO 450
0388          CALL ERROR(5)
0389          OPRND=0
0390    450   IF((OPRND.GE.0).AND.(OPRND.LE.15)) GOTO 460
0391          CALL ERROR(2)
0392          OPRND=0
0393    460   CALL MERGE(MASKA(OPRND,15),0)
0394          GOTO 700
```

```
        C
        C
        C PROCESS EXTENDED DATA STATEMENTS
0395    465   EXFLG=.TRUE.
        C PROCESS DATA STATEMENTS
0396    470   D=0
0397    475   CALL INTERP(OPRND,TYPE)
0398          D=D+1
0399          IF(TYPE.NE.0) GOTO 480
0400          CALL ERROR(5)
0401          OPRND=0
0402    480   IF((OPRND/2.GT.-32768).OR.(OPRND/2.LE.32767)) GO TO 490
0403          CALL ERROR(2)
0404          OPRND=0
0405    490   CODE=OPRND
0406          IF(D.EQ.1) WRITE(NWRITE,710) LN,IDINT(LOCCTR),CODE,READIN
0407          IF(D.NE.1) WRITE(NWRITE,495) LN,IDINT(LOCCTR),CODE
0408    495   FORMAT(1X,I4,3X,O6,3X,O6)
0409          CALL OUTCDE
0410          LOCCTR=LOCCTR+1
0411          IF((EXFLG).AND.(I.LE.78)) GOTO 475
0412          EXFLG=.FALSE.
0413          GOTO 210
        C
        C PROCESS EQUIVALENCE STATEMENTS
0414    500   I=0
0415          CALL GETFLD(OPRND)
0416          CALL SEARCH(OPRND,TYPE)
0417          IF(TYPE.EQ.0) CALL ERROR(5)
0418          IF(LISFLG) WRITE(NWRITE,510) LN,IDINT(OPRND),READIN
0419    510   FORMAT(2X,I4,6X,O8,7X,20A4)
0420          GOTO 210
        C
        C  PROCESS RESERVE STORAGE STATEMENTS
0421    520   CALL INTERP(OPRND,TYPE)
0422          IF(TYPE.NE.0) GOTO 530
0423          CALL ERROR(5)
0424          OPRND=1
0425    530   IF(OPRND.GT.0) GOTO 540
0426          CALL ERROR(2)
0427          OPRND=0
0428    540   LOCCTR=LOCCTR+OPRND
0429          OPRNDN=IDINT(OPRND)
0430          DO 545 D=1,OPRNDN
0431          CALL OUTCDE
0432    545   CONTINUE
0433          GOTO 720
        C
        C PROCESS SKIP STATEMENTS
0434    550   CALL INTERP(OPRND,TYPE)
```

```
0435            IF(TYPE.NE.0) GOTO 560
0436            GOTO 210
0437      560   IF((OPRND.GT.0).AND.(OPRND.LT.60)) GOTO 570
0438            OPRND=0
0439      570   OPRNDN=IDINT(OPRND)
0440            DO 580 D=1,OPRNDN
0441            IF(LISFLG)WRITE(NWRITE,575)
0442      575   FORMAT(/)
0443      580   CONTINUE
0444            GOTO 210
          C
          C PROCESS .ORG AND $LOC STATEMENTS
0445      590   CALL INTERP(OPRND,TYPE)
0446            IF(TYPE.NE.0) GOTO 600
0447            CALL ERROR(6)
0448            OPRND=0
0449      600   IF(OPRND.GE.0) GOTO 610
0450            CALL ERROR(2)
0451            OPRND=0
0452      610   LOCCTR=OPRND
0453            GOTO 720
          C
          C PROCESS PAGE STATEMENTS
0454      620   IF(LISFLG)WRITE(NWRITE,630)
0455      630   FORMAT(1H1)
0456            GOTO 210
          C
          C LIST COMMENT LINES, OPTN STATEMENTS,AND ILLEGAL OPCODE LINES
0457      660   IF(LISFLG) WRITE(NWRITE,670) LN,READIN
0458      670   FORMAT(2X,I4,21X,20A4)
0459            GOTO 210
          C
          C LIST MACHINE INSTRUCTIONS, DATA STATEMENTS
0460      700   IF(LISFLG) WRITE(NWRITE,710) LN,IDINT(LOCCTR),CODE,READIN
0461      710   FORMAT(2X,I4,3X,O6,3X,O6,3X,20A4)
0462            CALL OUTCOE
0463            LOCCTR=LOCCTR+1
0464            GOTO 210
          C
          C LIST ORG,$LOC
0465      720   IF(LISFLG) WRITE(NWRITE,730) LN,IDINT(LOCCTR),READIN
0466      730   FORMAT(2X,I4,3X,O8,10X,20A4)
0467            GOTO 210
          C
          C
          C PROCESS END STATEMENTS
0468      800   IF(LISFLG) WRITE(NWRITE,670) LN,READIN
0469      805   WRITE(NWRITE,810)
0470      810   FORMAT(56X,19H**END OF PASS TWO**/)
          C
```

```
        C LIST NUMBER OF PASS TWO ERRORS AND WORDS OF CODE
0471            IF(CNT.EQ.0) WRITE(NWRITE,144)
0472            IF(CNT.NE.0) WRITE(NWRITE,146) CNT
0473            IF(WDCNT.EQ.0) WRITE(NWRITE,820)
0474    820     FORMAT(2X,17HNO CODE GENERATED/)
0475            IF(WDCNT.NE.0) WRITE(NWRITE,830) WDCNT,WDCNT
0476    830     FORMAT(2X,I 8,2H (06,29H OCT) WORDS OF CODE GENERATED/)
        C
0477            REWIND 1
        C
0478            WRITE(NWRITE,630)
        C  GO BACK TO EOF SENSE AT PASS ONE SETUP
0479             GO TO 1
0480            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	SCODE1	012446	2707	RW,I,CON,LCL
2	SPDATA	000240	80	RW,D,CON,LCL
3	SIDATA	001202	321	RW,D,CON,LCL
4	SVARS	003332	877	RW,D,CON,LCL
5	STEMPS	000010	4	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL
7	B	000006	3	RW,D,OVR,GBL
8	C	000004	2	RW,D,OVR,GBL
9	D	021674	4574	RW,D,OVR,GBL
10	E	000010	4	RW,D,OVR,GBL

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ASMFLG	L*2	4-000016	BLNK	R*8	4-003210	CHAR	I*4	4-000000	CNT	I*2	7-000002	CODE	I*2	8-000000
CODFLG	L*2	4-000010	D	I*2	4-003270	DATA	R*8	4-003140	DOLND	R*8	4-003260	END	R*8	4-003170
ENDTB	I*2	9-021670	ENDTB1	I*2	4-003306	EQU	R*8	4-003130	EXFLG	L*2	4-000020	FLG	L*2	4-000006
I	I*2	6-000120	J	I*2	4-003276	KT	I*2	4-003316	KT1	I*2	4-003320	LASTLN	I*2	4-003300
LISFLG	L*2	4-000012	LN	I*2	7-000000	LOC	R*8	4-003230	LOCCTR	R*8	10-000000	N	I*2	4-003326
NREAD	I*2	6-000122	NWRITE	I*2	6-000124	OPCODE	R*8	4-003110	OPRND	R*8	4-003060	OPRNDN	I*2	4-003330
OPTN	R*8	4-003200	ORG	R*8	4-003120	P	I*2	4-003272	PAGE	R*8	4-003160	PASMDE	I*2	7-000004
PNT	I*2	4-003302	PT	I*2	4-003322	PTEMP	I*2	4-003314	P1	I*2	4-003304	REG	I*2	4-003324
RES	R*8	4-003150	SKP	R*8	4-003220	STOFLG	L*2	4-000004	TABFLG	L*2	4-000014	TABLND	I*2	9-021672
TEMPA	R*8	4-003070	TEMPB	R*8	4-003100	TEMPC	I*2	4-003310	TEMPD	I*2	4-003312	TITL	R*8	4-003250
TYPE	I*2	4-003274	WDCNT	I*2	8-000002	XDATA	R*8	4-003240						

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
LISCNC	I*2	4-003034	000024	10	(10)
OPTBLA	R*8	4-000022	000650	212	(53)
OPTBLB	I*2	4-000672	000152	53	(53)
READIN	I*4	6-000000	000120	40	(20)
SYMTAB	R*8	9-000000	007740	2032	(508)
SYMTAD	I*2	4-001044	001770	508	(508)
SYMTBB	R*8	9-007740	007740	2032	(508)
SYMTBC	I*2	9-017700	001770	508	(508)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS

1	1-000326	3	1-000374	4	1-000410	5	**	7'	3-000000
10	1-000556	20'	3-000112	22	1-000632	25'	3-000116	30	1-001116
35	1-001424	40	1-001472	45	1-001532	50	1-001556	60	1-001572
70	1-001634	75	1-001670	90	1-001734	100	1-001774	103	1-002016
105	1-002030	107	1-002054	108	1-002150	110	1-002166	120	1-002254
130	1-002312	132	1-002342	133	1-002400	135	1-002536	140	1-002554
142'	3-000174	144'	3-000226	146'	3-000246	148	1-002710	170	1-003124
180	**	185	**	193	1-003630	194	1-003712	195	1-003734
196'	3-000266	197'	3-000314	198	**	200'	3-000344	201	1-004266
210	1-004416	215	1-004510	220	1-004610	225	1-004650	230	1-004662
240	1-005120	250	1-005270	255	1-005554	260	1-005622	280	1-006106
283	1-006224	285	1-006242	295	1-006324	300	1-006346	305	1-006414
310	1-006462	315	1-006514	320	1-006556	322	1-006730	324	1-007012
325	1-007026	330	1-007070	340	1-007142	350	1-007164	352	1-007226
355	1-007260	360	1-007304	370	1-007350	390	1-007412	395	1-007514
400	1-007534	405	1-007546	410	1-007620	420	1-007652	430	1-007726
435	1-007774	440	1-010066	450	1-010130	460	1-010200	465	1-010242
470	1-010254	475	1-010266	480	1-010334	490	1-010414	495'	3-000436
500	1-010702	510'	3-000452	520	1-011042	530	1-011110	540	1-011142
545	**	550	1-011246	560	1-011276	570	1-011336	575'	3-000470
580	**	590	1-011440	600	1-011502	610	1-011534	620	1-011556
630'	3-000472	660	1-011616	670'	3-000476	700	1-011676	710'	3-000510
720	1-012032	730'	3-000532	800	1-012130	805	1-012204	810'	3-000550
820'	3-000602	830'	3-000632						

FUNCTIONS AND SUBROUTINES REFERENCED

ERRCLR	ERROR	FETCH	GETAN	GETFLD	GETLNG	GETRN	GETXN	INTERP	MASKA	MERGE	OUTCDE	RSMBL	SEARCH	SMBL	STORE
TAB	$IDINT	$INT													

TOTAL SPACE ALLOCATED = 041516 8615

```
0001            SUBROUTINE INTERP(OP,TYP)
        C INTERP SCANS TO THE FIRST CHAR OF AN OPERAND FIELD, AND INTERPRETS IT
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,FMT,CHAR1,CHAR2,ZCHAR,T,MAX
0004            LOGICAL FLG,SUBFLG,HEXFLG
0005            DOUBLE PRECISION OP,TEMP,LOCCTR,BLNK,RTMP
0006            REAL XCHAR
0007            DIMENSION FMT(3)
0008            COMMON/A/READIN(20),I,NREAD,NWRITE
0009            COMMON/E/LOCCTR
0010            DATA BLNK/8H        /
0011            TYP=1
        C TYP=0 IF UNDEFINED SYMBOLIC TERM, 1 IF DEFINED, OR 2 IF NUMERIC
0012            OP=0
0013            J=0
0014            FLG=.FALSE.
0015            CALL TAB
0016    1010    I=I+J
0017            J=0
0018            SUBFLG=.FALSE.
0019            HEXFLG=.FALSE.
0020            CALL FETCH(I,1,CHAR1)
        C BLANK OR COMMA IS DELIMITER
0021            IF(CHAR1.EQ.1H ) RETURN
0022            IF(CHAR1.NE.1H,) GOTO 1015
0023            IF(FLG) RETURN
0024            I=I+1
0025            GOTO 1010
        C IF NEGATIVE SIGNED TERM, SET SUBFLG
0026    1015    IF(CHAR1.EQ.1H-) SUBFLG=.TRUE.
        C SIGN NOT PART OF OPERAND
0027            IF((CHAR1.EQ.1H+).OR.(CHAR1.EQ.1H-)) I=I+1
0028            IF(FLG) GOTO 1020
0029            FLG=.TRUE.
0030            GOTO 1030
0031    1020    IF(TYP.EQ.1) GOTO 1030
        C IF UNDEFINED TERM IN SYMBOLIC EXPRESSION, ERROR
0032            CALL ERROR(6)
0033            OP=BLNK
0034            RETURN
0035    1030    CALL FETCH(I,1,CHAR1)
0036            CALL FETCH(I+1,1,CHAR2)
0037            IF(CHAR1.EQ.1HX) GOTO 1072
        C NOT HEX
0038            T=1HI
0039            MAX=1H9
0040            IF(CHAR1.NE.1H0) GOTO 1045
0041            T=1H0
0042            MAX=1H7
0043            GOTO 1050
```

```
0044     1045  IF((CHAR1.LT.1H0).OR.(CHAR1.GT.1H9)) GOTO 1070
         C OCTAL
0045     1050  CALL FETCH(I+J,1,CHAR1)
0046           IF(((CHAR1.EQ.1H )).OR.(CHAR1.EQ.1H+)).OR.((CHAR1.EQ.1H-).OR.
              *(CHAR1.EQ.1H,))) GOTO 1100
0047           IF((CHAR1.GE.1H0).AND.(CHAR1.LE.MAX)) GOTO 1060
0048           I=I+J
         C INVALID CHARACTER
0049           CALL ERROR(1)
0050           GOTO 1115
0051     1060  J=J+1
0052           IF((J.LE.10).AND.(I.LE.79).AND.((I+J).LE.79)) GOTO 1050
0053           I=I+J
         C TOO MANY DIGITS FOR REFORMATTING
0054           CALL ERROR(14)
0055           GOTO 1115
         C DOLLAR IS CURRENT LOCATION COUNTER VALUE
         C NOT NUMERIC
0056     1070  IF ( CHAR1.NE.1H$) GOTO 1090
0057           TEMP=LOCCTR
0058           TYP=1
0059           I=I+1
0060           GOTO 1110
         C NOT DECIMAL OR OCTAL
0061     1072  IF(CHAR2.NE.1H') GOTO 1090
0062           T=1HZ
         C HEXIDECIMAL NUMBER
0063           I=I+2
0064     1075  CALL FETCH(I+J,1,CHAR1)
0065           IF(CHAR1.NE.1H') GOTO 1077
0066           HEXFLG=.TRUE.
0067           GOTO 1100
0068     1077  IF(((CHAR1.GE.1H0).AND.(CHAR1.LE.1H9)).OR.((CHAR1.GE.1HA).AND.
              *(CHAR1.LE.1HF)))GOTO1080
0069           I=I+J
         C INVALID CHARACTERS IN HEX NUMBER- ERROR
0070           CALL ERROR(1)
0071           GOTO 1115
0072     1080  J=J+1
0073           IF((I+J.LE.80).AND.(J.LE.8)) GOTO 1075
0074           I=I+J
         C TOO MANY DIGITS IN HEX NUMBER- ERROR
0075           CALL ERROR(14)
0076           GOTO 1115
         C SYMBOLIC TERM
0077     1090  CALL GETFLD(TEMP)
0078           IF(TEMP.NE.BLNK) GOTO 1095
         C FIELD MISSING- ERROR
0079           CALL ERROR(7)
0080           GOTO 1115
```

```
        C SEARCH SYMBOL TABLE
0081    1095  CALL SEARCH(TEMP,TYP)
0082          IF(TYP.NE.0) GOTO 1110
        C UNDEFINED TERM
0083          OP=TEMP
0084          TYP=0
0085          GOTO 1010
        C GENERATE FOMRAT STATEMENT
0086    1100  IF(T.EQ.1HZ) GO TO 3000
0087          ENCODE(9,1105,FMT) I,T,J
        C REFORMAT NUMERIC DATA
        C REFORMAT TO INTEGER TYPE
0088          DECODE(80,FMT,READIN) ITMP
0089    1105  FORMAT(1H(,I2,2HX,A1,I2,1H))
0090    1106  TEMP=ITMP
        C IF HEX NUMBER WAS PROCESSED, SKIP TRAILING APOSTROPHE
0091          IF(HEXFLG) I=I+1
        C IF NEGATIVE TERM, SUBTRACT
0092    1110  IF(SUBFLG) TEMP= -TEMP
0093          OP=OP+TEMP
        C GO BACK TO LOOK FOR MORE TERMS
0094          GOTO 1010
0095    3000  RITMP=0.
0096          DO 3100 II=1,J
0097          J1=II-1
0098          CALL FETCH(I+J1,1,ZCHAR)
0099          IF(ZCHAR.EQ.1H0) XCHAR=0.
0100          IF(ZCHAR.EQ.1H1) XCHAR=1.
0101          IF(ZCHAR.EQ.1H2) XCHAR=2.
0102          IF(ZCHAR.EQ.1H3) XCHAR=3.
0103          IF(ZCHAR.EQ.1H4) XCHAR=4.
0104          IF(ZCHAR.EQ.1H5) XCHAR=5.
0105          IF(ZCHAR.EQ.1H6) XCHAR=6.
0106          IF(ZCHAR.EQ.1H7) XCHAR=7.
0107          IF(ZCHAR.EQ.1H8) XCHAR=8.
0108          IF(ZCHAR.EQ.1H9) XCHAR=9.
0109          IF(ZCHAR.EQ.1HA) XCHAR=10.
0110          IF(ZCHAR.EQ.1HB) XCHAR=11.
0111          IF(ZCHAR.EQ.1HC) XCHAR=12.
0112          IF(ZCHAR.EQ.1HD) XCHAR=13.
0113          IF(ZCHAR.EQ.1HE) XCHAR=14.
0114          IF(ZCHAR.EQ.1HF) XCHAR=15.
0115    3100  RITMP=RITMP+XCHAR*16.**(J-II)
0116          IF(RITMP.LT.32768.) GO TO 3200
0117          ITMP=RITMP-32768.
0118          ITMP=IOR(ITMP,"100000)
0119          GO TO 1106
0120    3200  ITMP=RITMP
0121          GO TO 1106
        C ERROR RETURN- CLEAR OPERAND VALUE
```

```
0122     1115  OP=0
0123           RETURN
0124           END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	003206	855	RW,I,CON,LCL
2	$PDATA	000020	8	RW,D,CON,LCL
3	$IDATA	000114	38	RW,D,CON,LCL
4	$VARS	000112	37	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL
7	E	000010	4	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
INTERP		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
BLNK	R*8	4-000056	CHAR1	I*4	4-000014	CHAR2	I*4	4-000020	FLG	L*2	4-000040	HEXFLG	L*2	4-000044
I	I*2	6-000120	II	I*2	4-000106	ITMP	I*2	4-000104	J	I*2	4-000102	J1	I*2	4-000110
LOCCTR	R*8	7-000000	MAX	I*4	4-000034	NREAD	I*2	6-000122	NWRITE	I*2	6-000124	OP	R*8	F-000002*
RITMP	R*8	4-000066	SUBFLG	L*2	4-000042	T	I*4	4-000030	TEMP	R*8	4-000046	TYP	I*2	F-000004*
XCHAR	R*4	4-000074	ZCHAR	I*4	4-000024									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FMT	I*4	4-000000	000014	6	(3)
READIN	I*4	4-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1010	1-000062	1015	1-000206	1020	1-000320	1030	1-000366	1045	1-000570
1050	1-000652	1060	1-001100	1070	1-001176	1072	1-001260	1075	1-001330
1077	1-001424	1080	1-001576	1090	1-001664	1095	1-001744	1100	1-002024
1105*	3-000000	1106	1-002166	1110	1-002222	1115	1-003170	3000	1-002266
3100	**	3200	1-003142						

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR FETCH GETFLD SEARCH TAB

TOTAL SPACE ALLOCATED = 003612 965

```
0001            SUBROUTINE GETFLD(FIELD)
        C GETFLD SCANS TO THE START OF A FIELD AND PICKS UP TO 8 CHARS FROM THE
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,FMT
0004            DOUBLE PRECISION FIELD,BLNK
0005            COMMON/A/READIN(20),I,NREAD,NWRITE
0006            DIMENSION FMT(2)
0007            DATA BLNK /8H        /
        C CLEAR FIELD
0008            FIELD=BLNK
        C SCAN TO FIRST CHARACTER OF FIELD
0009            CALL TAB
        C FIND LENGTH OF FIELD
0010            CALL GETLNG(J)
0011            K=J
0012            IF((I+J).GT.80) J=80-I
0013            IF(J.LT.1) RETURN
0014            IF(J.LE.8) GOTO 1119
0015            I=I+8
        C TOO MANY CHARACTERS--ERROR
0016            CALL ERROR(3)
0017            I=I-8
0018            J=8
        C GENERATE FORMAT STATEMENT
0019    1119    IF(I.EQ.0) GO TO 2000
0020            ENCODE(8,1120,FMT) I,J
0021            GO TO 3000
0022    2000    ENCODE(8,2120,FMT) J
0023    2120    FORMAT(2H(A,I1,5H     ))
0024    1120    FORMAT(1H(,I2,3HX,A,I1,1H))
        C TRANSFER CHARACTERS
0025    3000    DECODE(80,FMT,READIN) FIELD
        C UPDATE SCAN POINTER
0026            I=I+K
0027            RETURN
0028            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000402	129	RW,I,CON,LCL
2	$PDATA	000004	2	RW,D,CON,LCL
3	$IDATA	000046	19	RW,D,CON,LCL
4	$VARS	000024	10	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
GETFLD		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
BLNK	R*8	4-000010	FIELD	R*8	F-000002*	I	I*2	6-000120	J	I*2	4-000020	K	I*2	4-000022
NREAD	I*2	6-000122	NWRITE	I*2	6-000124									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FMT	I*4	4-000000	000010	4	(2)
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1119	1-000200	1120*	3-000016	2000	1-000262	2120*	3-000000	3000	1-000324

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR GETLNG TAB

TOTAL SPACE ALLOCATED = 000626 203

```
0001            SUBROUTINE FETCH(SKIP,TAKE,DEST)
        C FETCH PICKS UP TO 4 CHARACTERS FROM THE READIN BUFFER
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,FMT,DEST
0004            COMMON/A/ READIN(20),I,NREAD,NWRITE
0005            DIMENSION FMT(2)
        C CHECK FOR END OF BUFFER
0006            IF(SKIP.GT.79) SKIP=79
        C CHECK LENGTH OF FIELD
0007            IF((SKIP+TAKE).GT.80) TAKE=80-SKIP
0008            IF(TAKE.LT.1) RETURN
0009            IF(TAKE.GT.4) TAKE=4
        C GENERATE FORMAT STATEMENT
0010            IF (SKIP.EQ.0) GO TO 100
0011            ENCODE(8,1120,FMT) SKIP,TAKE
0012    1120    FORMAT(1H(,I2,3HX,A,I1,1H))
0013            GO TO 200
0014    100     ENCODE(8,2120,FMT) TAKE
0015    2120    FORMAT(2H(A,I1,5H    ))
        C TRANSFER CHARACTERS
0016    200     DECODE(80,FMT,READIN) DEST
0017            RETURN
0018            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000310	100	RW,I,CON,LCL
3	$IDATA	000034	14	RW,D,CON,LCL
4	$VARS	000010	4	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
FETCH		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
DEST	I*4	F-000006*	I	I*2	6-000120	NREAD	I*2	6-000122	NWRITE	I*2	6-000124	SKIP	I*2	F-000002*
TAKE	I*2	F-000004*												

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FMT	I*4	4-000000	000010	4	(2)
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	1-000176	200	1-000240	1120'	3-000000	2120'	3-000016		

TOTAL SPACE ALLOCATED = 000502 161

NO FPP INSTRUCTIONS GENERATED

```
0001          SUBROUTINE TAB
              C TAB SCANS TO THE NEXR NON-BLANK CHAR IN THE READIN BUFFERC
              C IF IT IS ALREADY AT A NON-BLANK CHARACTER, NO ACTION OCCURS
0002          IMPLICIT INTEGER(A-Z)
0003          INTEGER*4 READIN,CHAR
0004          COMMON/A/READIN(20),I,NREAD,NWRITE
0005    1130  CALL FETCH(I,1,CHAR)
0006           IF(CHAR.NE.1H .AND.CHAR.NE.1H,) RETURN
0007          I=I+1
0008          IF(I.GE.80) RETURN
0009          GOTO  1130
0010          END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000112	37	RW,I,CON,LCL
2	$PDATA	000004	2	RW,D,CON,LCL
3	$ICATA	000010	4	RW,D,CON,LCL
4	$VARS	000004	2	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
TAB		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CHAR	I*4	4-000000	I	I*2	6-000120	NREAD	I*2	6-000122	NWRITE	I*2	6-000124			

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1130	1-000012								

FUNCTIONS AND SUBROUTINES REFERENCED

FETCH

TOTAL SPACE ALLOCATED = 000260 88

NO FPP INSTRUCTIONS GENERATED

```
0001          SUBROUTINE GETLNG(J)
      C GETLNG FINDS THE NUMBER OF CHARS IN A FIELD
      C IT SHOULD BE CALLED WITH I POINTING TO THE FIRST CHAR ON THE FIELD
0002          IMPLICIT INTEGER (A-Z)
0003          INTEGER*4 READIN,CHAR
0004          COMMON/A/ READIN(20),I,NREAD,NWRITE
0005          J=0
0006    1140  CALL FETCH(I+J,1,CHAR)
0007          IF((CHAR.EQ.1H ).OR.(CHAR.EQ.1H+).OR.(CHAR.EQ.1H-).OR.
             *(CHAR.EQ.1H,)) RETURN
0008          J=J+1
0009          IF(I+J.LE.80) GOTO 1140
0010          RETURN
0011          END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000206	67	RW,I,CON,LCL
2	$PDATA	000004	2	RW,D,CON,LCL
3	$IDATA	000010	4	RW,D,CON,LCL
4	$VARS	000004	2	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
GETLNG		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CHAR	I*4	4-000000	I	I*2	6-000120	J	I*2	F-000002*	NREAD	I*2	6-000122	NWRITE	I*2	6-000124

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
READIN	I*4	6-000000	000120	40 (20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1140	1-000026								

FUNCTIONS AND SUBROUTINES REFERENCED

FETCH

TOTAL SPACE ALLOCATED = 000354 118

NO FPP INSTRUCTIONS GENERATED

```
0001            SUBROUTINE STORE(ARG,FLG,P)
          C STORE SAVES SYMBOL VALUES, AND SSSOCIATED FLAGS, IN THE SYMBOL TABLE
          C IT ALSO INCREMENTS THE POINTER
0002            IMPLICIT INTEGER(A-Z)
0003            DOUBLE PRECISION SYMTAB,SYMTBB,ARG
0004            COMMON/D/SYMTAB(508),SYMTBB(508),SYMTBC(508),END,TABLND
0005            SYMTBB(P)=ARG
0006            SYMTBC(P)=FLG
0007            P=P+1
0008            RETURN
0009            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000066	27	RW,I,CON,LCL
6	D	021674	4574	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
STORE		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ARG	R*8	F-000002*	END	I*2	6-021670	FLG	I*2	F-000004*	P	I*2	F-000006*	TABLND	I*2	6-021672

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
SYMTAB	R*8	6-000000	007740	2032	(508)
SYMTAB	R*8	6-007740	007740	2032	(508)
SYMTBC	I*2	6-017700	001770	508	(508)

TOTAL SPACE ALLOCATED = 021762 4601

```
0001            SUBROUTINE SEARCH(ARG,FLG)
        C SEARCH SEARCHES THE SYMBOL TABLE FOR THE CHARACTER STRING IN ARG
        C IF THE SYMBOL IS FOUND, THE VALUE IS RETURNED IN ARG, AND THE FLAG IN
        C IF THE SYMBOL IS NOT FOUND, A ZERO IS RETURNED FOR BOTH
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN
0004            DOUBLE PRECISION SYMTAB,SYMTBB,ARG
0005            COMMON/A/READIN(20),I,NREAD,NWRITE
0006            COMMON/D/SYMTAB(508),SYMTBB(508),SYMTBC(508),END,TABLND
0007            PT=0
0008            FLG=0
        C SEARCH THE SYMBOL TABLE
0009        1170  PT=PT+1
0010            IF(PT.EQ.END+1) PT=TABLND+1
        C SEARCH THE PREDEFINED SYMBOL TABLE
0011            IF(PT.GT.TABLND+8) GOTO 1175
0012            IF(ARG.NE.SYMTAB(PT)) GOTO 1170
0013            ARG=SYMTBB(PT)
        C IF DOUBLY DEFINED, ERROR
0014            IF(SYMTBC(PT).EQ.2) CALL ERROR(4)
0015            FLG=SYMTBC(PT)
0016            RETURN
0017        1175  ARG=0
0018            RETURN
0019            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000236	79	RW,I,CON,LCL
2	$PDATA	000004	2	RW,D,CON,LCL
3	$IDATA	000004	2	RW,D,CON,LCL
4	$VARS	000002	1	RW,D,CON,LCL
5	$TEMPS	000002	1	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL
7	D	021674	4574	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
SEARCH		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ARG	R*8	F-000002*	END	I*2	7-021670	FLG	I*2	F-000004*	I	I*2	6-000120	NREAD	I*2	6-000122
NWRITE	I*2	6-000124	PT	I*2	4-000000	TABLND	I*2	7-021672						

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
READIN	I*4	6-000000	000120	40	(20)
SYMTAB	R*8	7-000000	007740	2032	(508)
SYMTBB	R*8	7-007740	007740	2032	(508)
SYMTBC	I*2	7-017700	001770	508	(508)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1170	1-000032	1175	1-000220						

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR

TOTAL SPACE ALLOCATED = 022274 4702

```
0001          SUBROUTINE ERROR(NUM)
      C ERROR OUTPUTS ALL ERROR MESSAGES AND THE TITLES FOR ERROR MESSAGE LIST
      C IF PASMDE IS 1, LINE 1190 IS LISTED
      C IF PASMDE IS 2, ONLY ERROR MESSAGES ARE PRINTED
      C IF PASMDE EQUALS 3, LINES 1183 AND 1190 WILL BE PRINTED
0002          IMPLICIT INTEGER(A-Z)
0003          INTEGER*4 READIN
0004          COMMON/A/READIN(20),I,NREAD,NWRITE
0005          COMMON/B/LN,CNT,PASMDE
0006          COMMON/ERRCOM/IND(5),NUMTBL(5),OLDLNE,D
0007          IF(PASMDE.EQ.2) GOTO 1192
0008          IF(CNT.GT.0) GOTO 1185
0009          IF(PASMDE.EQ.3) GOTO 1182
0010          WRITE(NWRITE,1180)
0011    1180  FORMAT(56X,19H**PASS ONE ERRORS**)
0012          GOTO 1187
0013    1182  WRITE(NWRITE,1183)
0014    1183  FORMAT(56X,19H**PASS TWO ERRORS**)
0015    1185  IF(LN.EQ.OLDLNE) GOTO 1192
0016    1187  WRITE(NWRITE,1190) LN,READIN
0017    1190  FORMAT(/ 2X,18HFOUND AT LINE NUM.,I5,3H:  ,20A4)
0018          D=0
0019    1192  CNT=CNT+1
0020          D=D+1
0021          IF(D.GT.4) RETURN
0022          NUMTBL(D)=NUM
0023          IND(D)=I+26
0024          OLDLNE=LN
0025          RETURN
0026          ENTRY ERRGLR
0027          IF(D.EQ.0) RETURN
0028          IF(D.GT.4) D=4
0029          WRITE(NWRITE,1195)D,(IND(E),E=1,D)
0030    1195  FORMAT(2X,I2,5(I4,1H|))
0031          DO 1400 E=1,D
0032          GOTO(1201,1202,1203,1204,1205,1206,1207,1208,1209,1210,1211,1212,
             *1213,1214,1215)NUMTBL(E)
0033    1201  WRITE(NWRITE,1301)
0034          GOTO 1400
0035    1202  WRITE(NWRITE,1302)
0036          GOTO 1400
0037    1203  WRITE(NWRITE,1303)
0038          GOTO 1400
0039    1204  WRITE(NWRITE,1304)
0040          GOTO 1400
0041    1205  WRITE(NWRITE,1305)
0042          GOTO 1400
0043    1206  WRITE(NWRITE,1306)
0044          GOTO 1400
0045    1207  WRITE(NWRITE,1307)
```

```
0046            GOTO 1400
0047    1208    WRITE(NWRITE,1308)
0048            GOTO 1400
0049    1209    WRITE(NWRITE,1309)
0050            GOTO 1400
0051    1210    WRITE(NWRITE,1310)
0052            GOTO 1400
0053    1211    WRITE(NWRITE,1311)
0054            GOTO 1400
0055    1212    WRITE(NWRITE,1312)
0056            GOTO 1400
0057    1213    WRITE(NWRITE,1313)
0058            GOTO 1400
0059    1214    WRITE(NWRITE,1314)
0060            GOTO 1400
0061    1215    WRITE(NWRITE,1315)
0062    1400    CONTINUE
0063            D=0
0064            RETURN
0065    1301    FORMAT(2X,41HILLEGAL CHAR IN NUMERIC OPERAND--SET TO 0)
0066    1302    FORMAT(2X,42HVALUE OF OPERAND IS OUT OF RANGE--SET TO 0)
0067    1303    FORMAT(2X,37HSYMBOL TOO LONG--TRUNCATED TO 6 CHARS)
0068    1304    FORMAT(2X,42HSYMBOL DOUBLY DEFINED--FIRST VALUE ASSUMED)
0069    1305    FORMAT(2X,35HSYMBOL UNDEFINED AT END OF PASS ONE)
0070    1306    FORMAT(2X,45HUNDEFINED SYMBOL OR ILLEGAL FORWARD REFERENCE)
0071    1307    FORMAT(2X,31HMISSING OPERAND--VALUE SET TO 0)
0072    1308    FORMAT(2X,44HILLEGAL OR MISSING OPCODE--STATEMENT IGNORED)
0073    1309    FORMAT(2X,41HREQUIRED LABEL MISSING--STATEMENT IGNORED)
0074    1310    FORMAT(2X,34HWARNING:ODD-NUMBERED REGISTER USED)
0075    1311    FORMAT(2X,40HSYMBOL TABLE OVERFLOW--FATAL TO ASSEMBLY)
0076    1312    FORMAT(2X,43HWARNING:END DIRECTIVE MISSING--ASSUMED HERE)
0077    1313    FORMAT(2X,44HILLEGAL ADDRESSING MODE FOR THIS INSTRUCTION)
0078    1314    FORMAT(2X,41HTOO MANY DIGITS IN NUMERIC TERM--SET TO 0)
0079    1315    FORMAT(2X,42HINVALID REGISTER DESIGNATOR--A0/X1 ASSUMED)
0080            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	001312	357	RW,I,CON,LCL
2	$PDATA	000040	16	RW,D,CON,LCL
3	$IDATA	001424	394	RW,D,CON,LCL
4	$VARS	000002	1	RW,D,CON,LCL
5	$TEMPS	000002	1	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL
7	B	000006	3	RW,D,OVR,GBL
8	ERRCOM	000030	12	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ERRCLR		1-000300	ERROR		1-000000									

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CNT	I*2	7-000002	D	I*2	8-000026	E	I*2	4-000000	I	I*2	6-000120	LN	I*2	7-000000
NREAD	I*2	6-000122	NUM	I*2	F-000002*	NWRITE	I*2	6-000124	OLDLNE	I*2	8-000024	PASMDE	I*2	7-000004

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
IND	I*2	8-000000	000012	5	(5)
NUMTBL	I*2	8-000012	000012	5	(5)
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1180'	3-000000	1182	1-000072	1183'	3-000030	1185	1-000120	1187	1-000136
1190'	3-000060	1192	1-000210	1195'	3-000122	1201	1-000500	1202	1-000532
1203	1-000564	1204	1-000616	1205	1-000650	1206	1-000700	1207	1-000730
1208	1-000760	1209	1-001010	1210	1-001040	1211	1-001070	1212	1-001120
1213	1-001150	1214	1-001200	1215	1-001230	1301'	3-000140	1302'	3-000216
1303'	3-000276	1304'	3-000350	1305'	3-000430	1306'	3-000500	1307'	3-000562
1308'	3-000626	1309'	3-000710	1310'	3-000766	1311'	3-001036	1312'	3-001114
1313'	3-001174	1314'	3-001256	1315'	3-001334	1400	1-001256		

TOTAL SPACE ALLOCATED = 003166 827

NO FPP INSTRUCTIONS GENERATED

```
0001          SUBROUTINE SMBL
      C SMBL INTERPRETS THE OPERAND FOR A MACHINE INSTRUCTION WITH DIRECT ADDR
      C IT CHECKS THE RANGE, AND ASSEMBLES THE ADDRESS INTO THE CO&E
0002          IMPLICIT INTEGER(A-Z)
0003          DOUBLE PRECISION OPRND,LOCCTR,TEST
0004          COMMON/E/LOCCTR
0005          CALL INTERP(OPRND,TYPE)
0006          TEST=OPRND
0007          IF(TYPE.NE.0) GOTO 1415
0008          CALL ERROR(5)
0009          OPRND=0
0010          GOTO 1420
0011          ENTRY RSMBL
      C RSMBL INTERPRETS, CHECKS, AND ASSEMBLES ADDRESSES FOR RELATIVE OPERAND
0012          CALL INTERP(OPRND,TYPE)
0013          IF(TYPE.NE.0) GOTO 1410
0014          CALL ERROR(5)
0015          OPRND=0
0016          GOTO 1420
0017    1410  OPRND=OPRND-(LOCCTR+1)
0018          TEST=OPRND+128
0019    1415  IF((TEST.GE.0).AND.(TEST.LE.255)) GOTO 1420
0020          CALL ERROR(2)
0021          OPRND=0
0022    1420  CALL MERGE((MASKA(OPRND,255)),0)
0023          RETURN
0024          END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000316	103	RW,I,CON,LCL
2	$PDATA	000020	8	RW,D,CON,LCL
3	$IDATA	000032	13	RW,D,CON,LCL
4	$VARS	000022	9	RW,D,CON,LCL
6	E	000010	4	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
RSMBL		1-000072	SMBL		1-000000									

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
LOCCTR	R*8	6-000000	OPRND	R*8	4-000000	TEST	R*8	4-000010	TYPE	I*2	4-000020			

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1410	1-000144	1415	1-000206	1420	1-000256				

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR INTERP MASKA MERGE

TOTAL SPACE ALLOCATED = 000422 137

```
0001            SUBROUTINE GETRN(OPRND)
        C GETRN INTERPRETS A REGISTER DESIGNATOR WHEN ANY REGISTER MAY BE USED
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,CHAR
0004            DOUBLE PRECISION TEMP
0005            COMMON/A/READIN(20),I,NREAD,NWRITE
0006            CALL REGNME(OPRND,0,7)
0007            RETURN
0008            ENTRY GETAN(OPRND)
        C GETAN INTERPRETS A REGISTER DESIGNATOR WHEN ONLY AN ACUMULATOR DESIGNA
        C IS EXPECTED
0009            CALL REGNME(OPRND,0,3)
0010            RETURN
0011            ENTRY GETXN(OPRND)
        C GETXN INTERPRETS AN INDEX REGISTER DESIGNATOR WHEN ONLY THAT IS EXPECT
0012            CALL FETCH(I,1,CHAR)
0013            IF(CHAR.NE.1H,) GOTO 1430
0014            I=I+1
0015            CALL GETFLD(TEMP)
0016            CALL SEARCH(TEMP,TYPE)
0017            OPRND=TEMP
0018            IF(TYPE.NE.0) GOTO 1440
0019            CALL ERROR(15)
0020            OPRND=4
0021            GOTO 1440
0022    1430    OPRND=3
0023    1440    OPRND=OPRND-2
        C THE REGISTER DESIGNATOR CODE IS RETURNED IN A FORM READY FOR ASSEMBLY
0024            RETURN
0025            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000306	99	RW,I,CON,LCL
2	$PDATA	000024	10	RW,D,CON,LCL
3	$IDATA	000046	19	RW,D,CON,LCL
4	$VARS	000016	7	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
GETAN		1-000046	GETRN		1-000000	GETXN		1-000106						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CHAR	I*4	4-000000	I	I*2	6-000120	NREAD	I*2	6-000122	NWRITE	I*2	6-000124	OPRND	I*2	F-000002*
TEMP	R*8	4-000004	TYPE	I*2	4-000014									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1430	1-000254	1440	1-000270						

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR FETCH GETFLD REQNME SEARCH

TOTAL SPACE ALLOCATED = 000544 178

```
0001            SUBROUTINE REGNME(OPRND,L,U)
      C REGNME GETS A REGISTER DESIGNATOR, AND CHECKS TO SEE THAT IT DEFINES
      C REGISTER, IT IS CALLED BY GETRN
0002            IMPLICIT INTEGER(A-Z)
0003            INTEGER*4 READIN,CHAR
0004            DOUBLE PRECISION TEMP
0005            COMMON/A/READIN(20),I,NREAD,NWRITE
0006            CALL TAB
0007            CALL FETCH(I,1,CHAR)
0008            IF(CHAR.EQ.1H,) I=I+1
0009            CALL INTERP(TEMP,TYPE)
0010            OPRND=TEMP
0011            IF(TYPE.EQ.1) GOTO 1420
0012            CALL ERROR(15)
0013            OPRND=L
      C THE BOUNDS FOR VALID DESIGNATOR NUMBERS ARE GIVEN BY L AND U
0014    1420    IF((OPRND.GE.L).AND.(OPRND.LE.U)) RETURN
0015            CALL ERROR(2)
0016            OPRND=L
0017            RETURN
0018            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000230	76	RW,I,CON,LCL
2	$PDATA	000014	6	RW,D,CON,LCL
3	$IDATA	000030	12	RW,D,CON,LCL
4	$VAR8	000016	7	RW,D,CON,LCL
6	A	000126	43	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
REGNME		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CHAR	I*4	4-000000	I	I*2	6-000120	L	I*2	F-000004*	NREAD	I*2	6-000122	NWRITE	I*2	6-000124
OPRND	I*2	F-000002*	TEMP	R*8	4-000004	TYPE	I*2	4-000014	U	I*2	F-000006*			

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
READIN	I*4	6-000000	000120	40	(20)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
1420	1-000154								

FUNCTIONS AND SUBROUTINES REFERENCED

ERROR	FETCH	INTERP	TAB

TOTAL SPACE ALLOCATED = 000440 144

```
0001            SUBROUTINE OUTCDE
          C OUTCDE OUTPUTS ONE WORD OF CODE, ANS INCREMENTS THE WORD COUNTER
0002            IMPLICIT INTEGER(A-Z)
0003            COMMON/C/CODE,WDCNT
0004            WDCNT=WDCNT+1
0005            RETURN
0006            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000030	12	RW,I,CON,LCL
6	C	000004	2	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
OUTCOE		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CODE	I*2	6-000000	WDCNT	I*2	6-000002									

TOTAL SPACE ALLOCATED = 000034 14

NO FPP INSTRUCTIONS GENERATED

```
0001            FUNCTION MASKA(OPRND,ARG)
                C MASKA PERFORMS BITWISE LOGICAL MASKAING
                C OPRND IS THE WORD TO BE MASKAED
                C ARG IS THE MASKA
                C MASKA IS THE INTEGER RESULT
0002            IMPLICIT INTEGER(A-Z)
0003            DOUBLE PRECISION OPRND
0004            MASKA=IAND(IDINT(OPRND),ARG)
0005            RETURN
0006            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000056	23	RW,I,CON,LCL
3	$IDATA	000004	3	RW,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
MASKA	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ARG	I*2	F-000004*	OPRND	R*8	F-000002*									

FUNCTIONS AND SUBROUTINES REFERENCED

$IDINT

TOTAL SPACE ALLOCATED = 000064 26

NO FPP INSTRUCTIONS GENERATED

```
0001            SUBROUTINE MERGE(ARG,SHFT)
                C MERGE PERFOMS A LEFT CIRCULAR SHIFT TO JUSTIFY AFIELD, AND THEN
                C A LOGICAL MERGE
0002            IMPLICIT INTEGER(A-Z)
0003            COMMON/C/ACCUM,WDCNT
0004            TEMP=ISHFT(ARG,SHFT)
0005            ACCUM=IOR(TEMP,ACCUM)
0006            RETURN
0007            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000064	26	RW,I,CON,LCL
3	$IDATA	000006	3	RW,D,CON,LCL
4	$VARS	000002	1	RW,D,CON,LCL
6	C	000004	2	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
MERGE		1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ACCUM	I*2	6-000000	ARG	I*2	F-000002*	SHFT	I*2	F-000004*	TEMP	I*2	4-000000	WDCNT	I*2	6-000002

FUNCTIONS AND SUBROUTINES REFERENCED

$ISHFT

TOTAL SPACE ALLOCATED = 000100 32

NO FPP INSTRUCTIONS GENERATED

CARDS,CARDS/-SP=CARDS/CO:20

APPENDIX C
ACE ALGORITHM

Following is a listing of the ACE Algorithm subprogram coded in FORTRAN as compiled by the CDC CYBER-74 series computer system.

```
      SUBROUTINE ACE                                                    000370
C                                                                       000380
C     AFAL ACE GENERATED FOR EXPECTED NUMBER OF HITS                    000390
C     SEE AFAL TR-73-20                                                 000400
C                                                                       000410
      COMMON/INPT/ AZ,EL,RANGE,IFLAG                                    000420
      COMMON/OUTPT/ EXPHTS                                              000430
      COMMON/CONST/ BULN,SIGT,SIGBS                                     000440
      COMMON/ACECOM/ SQ2PI,E,AZOLD,ELOLD,W2                             000450
      DATA SQ2PI/2.5066283/                                             000451
      DATA E/2.7182818/                                                 000452
C                                                                       000460
C-----INPUTS--------------                                              000470
C                                                                       000480
C     AZ              THE AZIMUTH POSITION OF THE COMPUTED              000490
C               BULLETS AT TARGET RANGE WITH RESPECT TO THE             000500
C               TARGET     (RAD)                                        000510
C     EL              THE ELEVATION POSITION OF THE COMPUTED            000520
C               BULLETS AT TARGET RANGE WITH RESPECT TO THE             000530
C               TARGET     (RAD)                                        000540
C     RANGE           RANGE OF THE TARGET  (FEET)                       000550
C     IFLAG           INITIALIZATION FLAG (IFLAG=1---RUN)               000560
C                                         (IFLAG.NE.1---INITIALIZE)     000570
C                                                                       000580
C                                                                       000590
C                                                                       000600
C-----OUTPUT--------------                                              000610
C     EXPHTS          EXPECTED NUMBER OF HITS PER COMPUTER CYCLE        000620
C                                                                       000640
C                                                                       000650
C                                                                       000660
C-----INPUT CONSTANTS-----------                                        000670
C     BULN            THE NUMBER OF BULLETS AT TARGET RANGE           000680
C               PER COMPUTER CYCLE TIME  (FIRE RATE * CYCLE TIME)       000690
C     SIGT            STANDARD DEVIATION OF THE TARGET  (FEET)        000700
C     SIGBS           STANDARD DEVIATION OF THE BULLET                000710
C               STREAM OR BULLET DISPERSION (RAD)                       000720
C                                                                       000750
C                                                                       000760
C                                                                       000770
C-----INTERNAL CONSTANTS-------                                         000780
C     SQ2PI           SQUARE ROOT OF (2.*3.1415926536)                000810
C     E               EXPO. NUMBER  (2.7182818285)                    000820
C     AZOLD           PREVIOUS VALUE OF AZIMUTH POSITION (RAD)        000830
C     E'OLD           PREVIOUS VALUE OF ELEVATION POSITION (RAD)      000840
C     W2              SIGBS*SIGBS  (RAD**2)                           000850
C                                                                       000860
C                                                                       000870
C                                                                       000880
      IF(IFLAG.EQ.1) GO TO 50                                           000890
C                                                                       000900
C     DEFINE INTERNAL CONSTANTS                                         000910
C                                                                       000920
      W2=SIGBS*SIGBS                                                    000980
      AZOLD=AZ                                                          000990
      ELOLD=EL                                                          001000
C                                                                       001010
```

```
      C     COMPLETE INITIALIZATION                                          001020
      C                                                                      001030
   60         RETURN                                                        001040
      C                                                                      001050
      C     BEGIN MAIN LOOP TO COMPUTE EXPHTS                                001060
      C                                                                      001070
      C     COMPUTE RELATIVE MOTION VECTOR                                   001080
   65 C                                                                      001090
      50     AZL=AZOLD-AZ                                                   001100
             ELL=ELOLD-EL                                                   001110
             XMAGL2=AZL*AZL+ELL*ELL                                         001120
             XMAGL=SQRT(XMAGL2)                                             001130
   70 C                                                                      001140
      C     COMPUTE ANGLE                                                    001150
      C                                                                      001160
             XNUM=AZL*AZOLD+ELL*ELOLD                                       001170
             RMAG2=AZOLD*AZOLD+ELOLD*ELOLD                                  001180
   75        RMAG=SQRT(RMAG2)                                               001190
             IF(XMAGL.EQ.0.) XMAGL=.000001                                  001200
             IF(RMAG.EQ.0.) RMAG=.000001                                    001210
             XDEM=XMAGL*RMAG                                                001220
             COSDL=XNUM/XDEM                                                001230
   80        IF(COSDL.GT.1.) COSDL=1.                                       001232
             IF(COSDL.LT.-1.) COSDL=-1.                                     001234
      C                                                                      001240
      C     COMPUTE MISS VECTOR OF TARGET WITH RESPECT TO RELATIVE MOTION    001250
      C     VECTOR                                                           001251
   85 C                                                                      001260
             SINDL2=1.-COSDL*COSDL                                          001270
             SINDL=SQRT(SINDL2)                                             001280
             XT=RMAG*SINDL                                                  001290
             YT=RMAG*COSDL-XMAGL/2.                                         001300
   90        ST=SIGT/RANGE                                                  001310
             S2=ST*ST                                                       001320
      C                                                                      001330
      C     BEGIN COMPUTATION OF EXPECTED NUMBER OF HITS                     001340
      C                                                                      001350
   95        SW=S2+W2                                                       001360
             EE=-(XT*XT)/(2.*SW)                                            001370
             XNUM=BULN*S2*SQ2PI*E**EE                                       001380
             XDEM=XMAGL*SQRT(SW)                                            001390
             ERPL=(XMAGL/2.-YT)/ST                                          001400
  100        ERMI=(-XMAGL/2.-YT)/ST                                         001410
      C                                                                      001440
      C     COMPUTE EXPECTED NUMBER OF HITS                                  001450
      C                                                                      001460
             EXPHTS=XNUM*(ERF(ERPL)-ERF(ERMI))/XDEM                         001470
  105 C                                                                      001480
      C     SAVE AZ AND EL FOR NEXT COMPUTATION                              001490
      C                                                                      001500
             AZOLD=AZ                                                       001510
             ELOLD=EL                                                       001520
  110        RETURN                                                         001530
             END                                                            001540
```

```
      FUNCTION ERF(X)                                                   001550
C                                                                       001560
C     ERROR FUNCTION CURVE FIT TO A THIRD DEGREE EQUATION               001570
C     OVER THE INTERVAL -3.5 TO 3.5                                     001580
C     OUTSIDE THIS INTERVAL  ERF ASSIGNED EITHER -.5 OR .5              001590
C                                                                       001600
      LOGICAL NEG                                                       001610
      NEG=.FALSE.                                                       001620
      IF(X.LT.0.) NEG=.TRUE.                                            001630
      X=ABS(X)                                                          001640
      IF(X.GE.3.5) GO TO 100                                            001650
      IF(X.LE..2) GO TO 50                                              001660
      ERF=-.01322336+.49613046*X-.15942666*X*X+.01698544*X*X*X          001670
      GO TO 150                                                         001680
50    ERF=.39629855*X                                                   001690
      GO TO 150                                                         001700
100   ERF=.5                                                            001710
150   IF(NEG) ERF=-ERF                                                  001720
      RETURN                                                            001730
      END                                                               001740
```

APPENDIX D

LAMARS SUPPORT PROGRAMS

Following are the listings of the five subprograms as compiled by the CDC CYBER-74 series computer system for the LAMARS fire control integration to be implemented on the ROLM 16/64 computer.

```
      SUBROUTINE RNGFL
C*KALMAN FILTER--CONSTANT ACCELERATION MODEL
C
      COMMON/GMODE/MODEWORD(10),MODEOPT(10),MODEERR(10),SWBIT(2:16)
            LOGICAL SWBIT
         EQUIVALENCE (MODEWORD(1),MODE   )
      COMMON/GDATA/ INTIN(50),FPIN(50),INTOUT(50),FPOUT(100)
         EQUIVALENCE (FPIN(20),RDRRNG)                    :RADAR RANGE
      COMMON/GVARB/ IVAR(50),FVAR(50),ICON(50),FCON(50)
         EQUIVALENCE (FVAR(1),DTHIGH)          :DT FOR HIGH RATE
         EQUIVALENCE (FVAR(2),DTMED )          :DT FOR MED RATE
         EQUIVALENCE (FVAR(3),DTLOW )          :DT FOR LOW RATE
         EQUIVALENCE (FVAR(4),RTALEX)          :RANGE EST FROM FILTER
         EQUIVALENCE (FVAR(5),VTA_E(1))        :VEL TGT/ATT EST LOS C.S.
            DIMENSION VTALE(3)
            EQUIVALENCE (VTALE(1),VTALEX)
            EQUIVALENCE (VTALE(2),VTALEY)
            EQUIVALENCE (VTALE(3),VTALEZ)
C*INPUT REQUIRED!   T= SAMPLING RATE,SEC.
C                SVSQ=VARIANCE,MEASUREMENT NOISE
C                SWSQ=VARIANCE,MANUEVER NOISE
C                   Z=MEASUREMENT,FEET
      COMMON/_RNGF/ ZZ(2),XH(3),P(3,3),XHP(3),PPP(3,3),TEM(3,3),PP(3,3),
     1          PHI(3,3),S(3,3),C(3),TM(3),KK,XK(3),T,T2,T3,T4,SVSQ,SWSQ
      DATA ((PHI(I,J),I=1,3),J=1,3)/1.,0.,0.,0.,1.,0.,0.,0.,1./
      DATA SVSQ,SWSQ/1000.,80000./
C
      Z=RDRRNG
      IF(MODE.GE.5) GO TO 30
      T=DTHIGH
      DO 22 J=1,3
      XH(J)=0.
   22 XK(J)=0.
      T2=T*T
      T3=T*T2
      T4=T*T3
      TEMP=SWSQ*T2/36.0
      S(1,1)=TEMP*T4
      S(2,1)=TEMP*3.*T3
      S(3,1)=TEMP*6.*T2
      S(1,2)=S(2,1)
      S(2,2)=TEMP*9.*T2
      S(3,2)=TEMP*18.*T
      S(1,3)=S(3,1)
      S(2,3)=S(3,2)
      S(3,3)=TEMP*36.
      PHI(1,2)=T
      PHI(2,3)=T
      PHI(1,3)=T2/2.
      KK=0
      MODE=MODE+5
   30 KK=KK+1
      IF(KK-3)35,37,40
   35 ZZ(KK)=Z
      IF(KK.EQ.2) MODE=MODE+5
      RETURN
   37 XH(1)=Z
```

```
                      XH(2)=(3.*Z-4.*ZZ(2)+ZZ(1))/(2.*T)
                      XH(3)=0.
   60                 P(1,1)=SVSQ
                      P(2,1)=3.*SVSQ/(2.*T)
                      P(3,1)=SVSQ/T2
                      P(1,2)=P(2,1)
                      P(2,2)=13.*SVSQ/(2.*T2)
   65                 P(3,2)=6.*SVSQ/T3
                      P(1,3)=P(3,1)
                      P(2,3)=P(3,2)
                      P(3,3)=6.*SVSQ/T4+SWSQ
                      GO TO 99
   70           40    DO 42 J=1,3
                      XHP(J)=XH(J)
                      DO 42 I=1,3
                42    PPP(I,J)=P(I,J)
                C*FILTER EQ 1--P(K/K-1)
   75                 DO 50 J=1,3
                      DO 50 I=1,3
                50    TEM(I,J)=PPP(I,1)*PHI(J,1)+PPP(I,2)*PHI(J,2)+PPP(I,3)*PHI(J,3)
                      DO 60 J=1,3
                      DO 60 I=1,3
   80           60    PP(I,J)=PHI(I,1)*TEM(1,J)+PHI(I,2)*TEM(2,J)+PHI(I,3)*TEM(3,J)+
                     1       S(I,J)
                C*FILTER EQ 2--KALMAN GAIN
                      TEMPX=1.0/(PP(1,1)+SVSQ)
                      DO 70 J=1,3
   85           70    XK(J)=TEMPX*PP(1,J)
                C*FILTER EQ 3--ESTIMATES
                      DO 80 J=1,3
                80    C(J)=PHI(J,1)*XHP(1)+PHI(J,2)*XHP(2)+PHI(J,3)*XHP(3)
                      SCALAR=C(1)
   90                 TEMPX=Z-SCALAR
                      DO 85 J=1,3
                85    TM(J)=XK(J)*TEMPX
                      DO 88 J=1,3
                88    XH(J)=C(J)+TM(J)
   95           C*FILTER EQ 4--P(K/K)
                      TEM(1,1)=1.0-XK(1)
                      TEM(2,1)=-XK(2)
                      TEM(3,1)=-XK(3)
                      TEM(1,2)=0.
  100                 TEM(2,2)=1.
                      TEM(3,2)=0.
                      TEM(1,3)=0.
                      TEM(2,3)=0.
                      TEM(3,3)=1.
  105                 DO 90 J=1,3
                      DO 90 I=1,3
                90    P(I,J)=TEM(I,1)*PP(1,J)+TEM(I,2)*PP(2,J)+TEM(I,3)*PP(3,J)
                C
                  99      PTALEX=XH(1)
  110                     VTALEX=XH(2)
                      IF(MODE.LT.20) MODE=MODE+10
                      RETURN
                      END
```

```
      SUBROUTINE DIP1
C     AFAL DIRECTOR GUNSIGHT
C     SEE AFAL TR-75-52
      REAL KH
      COMMON/GMODE/MODEWORD(10),MODEOPT(10),MODEERR(10),SWBIT(2:16)
          LOGICAL SWBIT
         EQUIVALENCE (MODEWORD(7),MODE    )
      COMMON/GDNAV/ INTIN(50),FPIN(50),INTOUT(50),FPOUT(100)
         EQUIVALENCE (FPOUT(1),DATA84(1))          ;OUTPUT TO HUD
            DIMENSION DATA84(43)
         EQUIVALENCE (FPOUT(82),UNPABYD1 )         ;UN VEC PIP BRST CS D1
         EQUIVALENCE (FPOUT(83),UNPABZD1 )         ;UN VEC PIP BRST CS D1
      COMMON/GVARB/ IVAR(50),FVAR(50),ICON(50),FCON(50)
         EQUIVALENCE (FVAR(4),PTALEX)          ;RANGE EST FROM FILTER
         EQUIVALENCE (FVAR(11),OMLILE(1))      ;OMEGA LOS LOS C.S. EST
            DIMENSION OMLILE(3)
            EQUIVALENCE (OMLILE(1),OMLILEX)
            EQUIVALENCE (OMLILE(2),OMLILEY)
            EQUIVALENCE (OMLILE(3),OMLILEZ)
         EQUIVALENCE (FVAR(27),VF        )      ;BLT VEL USED BY LCOS
         EQUIVALENCE (FVAR(28),RTF       )      ;INVERSE TOF BY LCOS
         EQUIVALENCE (FVAR(30),ROC       )      ;CURV & ACCEL EFFECT
         EQUIVALENCE (FCON(20),RRH     )        ;INVERSE HARMON.RANGE
         EQUIVALENCE (FCON(23),PGHA(1))         ;POS GUN/HUD AC C. S.
            DIMENSION PGHA(3)
            EQUIVALENCE (PGHA(1),PGHAX)
            EQUIVALENCE (PGHA(2),PGHAY)
            EQUIVALENCE (PGHA(3),PGHAZ)
         EQUIVALENCE (FCON(26),XBIASHUD )       ;X-BIAS FOR HUD (RAD)
         EQUIVALENCE (FCON(27),YBIASHUD )       ;Y-BIAS FOR HUD (RAD)
         EQUIVALENCE (FCON(28),XSCALEHUD ) ;X-SCALE FAC FOR HUD (IN/RAD)
         EQUIVALENCE (FCON(29),YSCALEHUD ) ;Y-SCALE FAC FOR HUD (IN/RAD)
C
C-----INPUTS---------------
C     OMLILEY,OMLILEZ   Y AND Z COMPONENTS OF ANGULAR RATE OF LOS
C                        IN A NON-ROLL STABILIZED LOS COORDINATE SYSTEM
C                        (ROLL = 0 BETWEEN A/C AND LOS COORDINATES)
C     PTALEX            RANGE TO TARGET (FT)
C     RTF               RECIPROCAL OF BULLET TIME OF FLIGHT (1/SEC)
C     VF                AVERAGE BULLET SEPARATION SPEED (FT/SEC)
C     ROC               BALLISTIC CURVATURE AND ACCELERATION CORRECTION
C
C-----OUTPUTS----------
C     UNPABYD1,UNPABZD1       Y AND Z COMPONENTS OF PIPPER POSITION
C                             RELATIVE TO BORESIGHT
C
C-----CONSTANTS----------
C     PGHAY,PGHAZ       POSITION OF GUN RELATIVE TO HUD IN A/C COORDINAT
C     RRH               RECIPROCAL OF HARMONIZATION RANGE
C
C     COMPUTE SIN AND COS OF LEAD ANGLES
C
      IF(MODE.GE.10) GO TO 10
      MODE=10
      RETURN
10    CONTINUE
      KH=1.-PTALEX*RRH
```

```
                SLAD=(-PTALEX*OMLILEZ+KH*PGHAY*RTF)/VF
                CLAD=1.-0.5*SLAD**2
60              SLED=(-PTALEX*OMLILEY+RDC-KH*PGHAZ*RTF)/(CLAD*VF)
                CLED=1.-0.5*SLED**2
              C
              C     CONVERT FROM EULER ANGLES TO UNIT VECTOR
              C
65              UNPABYD1=CLED*SLAD
                UNPABZD1=-SLED
                DATA8+(+1)=(UNPABYD1+XBIASHUD)*XSCALEHUD
                DATA8+(+2)=(-UNPABZD1+YBIASHUD)*YSCALEHUD
                MODE=20
70              RETURN
                END
```

```
      SUBROUTINE DIR2
C     USAFA DIRECTOR GUNSIGHT
C     SEE USAFA-TR-74-17
      REAL LAMY,LAMZ
      COMMON/GMODE/MODEWORD(10),MODEOPT(10),MODEERR(10),SWBIT(2:16)
     1    LOGICAL SWBIT
        EQUIVALENCE (MODEWORD(8),MODE   )
      COMMON/GDMAT/ INTIN(50),FPIN(50),INTOUT(50),FPOUT(100)
        EQUIVALENCE (FPIN(17),SFAIA(1))              :ATT. SPEC. FORCE
        DIMENSION  SFAIA(3)
        EQUIVALENCE (SFAIA(1),AX)
        EQUIVALENCE (SFAIA(2),AY)
        EQUIVALENCE (SFAIA(3),AZ)
      EQUIVALENCE (FPOUT(1),DATA84(1))          :OUTPUT TO HUD
        DIMENSION DATA84(43)
      EQUIVALENCE (FPOUT(84),UNPABYD2 )         :UN VEC PIP BRST CS D2
      EQUIVALENCE (FPOUT(85),UNPABZD2 )         :UN VEC PIP BRST CS D2
      COMMON/GVARBL/ IVAR(50),FVAR(50),ICON(50),FCON(50)
      EQUIVALENCE (FVAR(4),PTALEX)          :RANGE EST FROM FILTER
      EQUIVALENCE (FVAR(8),UNTAGE(1))       :UN VEC TGT AC C.S. EST
        DIMENSION UNTAGE(3)
        EQUIVALENCE (UNTAGE(1),UNTAGEX)
        EQUIVALENCE (UNTAGE(2),UNTAGEY)
        EQUIVALENCE (UNTAGE(3),UNTAGEZ)
      EQUIVALENCE (FVAR(14),OMLIGE(1))      :OMEGA LOS GUN C.S. EST
        DIMENSION OMLIGE(3)
        EQUIVALENCE (OMLIGE(1),OMLIGEX)
        EQUIVALENCE (OMLIGE(2),OMLIGEY)
        EQUIVALENCE (OMLIGE(3),OMLIGEZ)
      EQUIVALENCE (FVAR(27),VF        )     :BLT VEL USED BY LCOS
      EQUIVALENCE (FVAR(28),RTF       )     :INVERSE TOF BY LCOS
      EQUIVALENCE (FVAR(29),BC        )     :BALLISTIC CURV BY LCOS
      EQUIVALENCE (FCON(20),RRH     )       :INVERSE HARMON.RANGE
      EQUIVALENCE (FCON(23),PGHA(1))        :POS GUN/HUD AC C. S.
        DIMENSION PGHA(3)
        EQUIVALENCE (PGHA(1),PGHAX)
        EQUIVALENCE (PGHA(2),PGHAY)
        EQUIVALENCE (PGHA(3),PGHAZ)
      EQUIVALENCE (FCON(26),XBIASHUD )      :X-BIAS FOR HUD (RAD)
      EQUIVALENCE (FCON(27),YBIASHUD )      :Y-BIAS FOR HUD (RAD)
      EQUIVALENCE (FCON(28),XSCALEHUD ) :X-SCALE FAC FOR HUD (IN/RAD)
      EQUIVALENCE (FCON(29),YSCALEHUD ) :Y-SCALE FAC FOR HUD (IN/RAD)
C
C-----INPUTS-----------
C     OMLIGEX,OMLIGEY,OMLIGEZ   ANGULAR RATE OF LOS IN GUN COORDINATES
C                                  (RAD/SEC)
C     UNTAGEX,UNTAGEY,UNTAGEZ      COMPONENTS OF LOS UNIT VECOR IN GUN
C                                    COORD.
C     AX,AZ              A/C ACCELEROMETER MEASUREMENTS (FT/SEC/SEC,
C                           POS FORWARD AND DOWN)
C     PTALEX              RANGE TO TARGET (FT)
C     RTF                RECIPROCAL OF BULLET TIME OF FLIGHT (1/SEC)
C     VF                 AVERAGE BULLET SEPARATION SPEED (FT/SEC)
C     BC                 BALLISTIC CURVATURE
C
C-----OUTPUTS----------
C     UNPABYD2,UNPABZD2       Y AND Z COMPONENTS OF PIPPER POSITION
```

```
C                          RELATIVE TO BORESIGHT
C
C-----CONSTANTS----------
C      PGHAY,PGHAZ      POSITION OF GUN RELATIVE TO HUD IN A/C COORDINATES
C      RRH              RECIPROCAL OF HARMONIZATION RANGE
C
       IF(MODE.GE.10) GO TO 10
       MODE=10
       RETURN
10     CONTINUE
       TF=1./PTF
       RRNGE=1./PTALEX
C
C      COMPUTE KINEMATIC LEAD AND BALLISTIC CURVATURE
C
       OMEG=OMLIGEX*UNTAGEX+OMLIGEY*UNTAGEY+OMLIGEZ*UNTAGEZ
       LAMY=TF*(OMLIGEY-OMEG*UNTAGEY)+BC/VF
     .   + TF*PTALEX*OMEG*(UNTAGEZ*OMLIGEX-UNTAGEX*OMLIGEZ)/(2.*VF)
     .   + TF*(UNTAGEZ*AX-UNTAGEX*AZ)/(2.*VF)
       LAMZ=TF*(OMLIGEZ-OMEG*UNTAGEZ)
     .   + TF*PTALEX*OMEG*(UNTAGEX*OMLIGEY-UNTAGEY*OMLIGEX)/(2.*VF)
     .   - TF*UNTAGEY*AX/(2.*VF)
C
C      ADD HARMONIZATION AND PARALLAX TERMS
C
       UNPARYD2=-LAMZ-(RRH-RRNGE)*PGHAY
       UNPARZD2=LAMY-(RRH-RRNGE)*PGHAZ
       DATA3+(+1)=(UNPARYD2+XBIASHUD)*XSCALEHUD
       DATA3+(+2)=(-UNPARZD2+YBIASHUD)*YSCALEHUD
       MODE=20
       RETURN
       END
```

```
      SUBROUTINE DIR3
C     AFAL 2ND ORDER DIRECTOR GUNSIGHT
C     CONTACT CAPT SILVERTHORN FOR DERIVATION
      COMMON/GMODE/MODEWORD(10),MODEOPT(10),MODEERR(10),SWBIT(2:16)
           LOGICAL SWBIT
          EQUIVALENCE (MODEWORD(9),MODE    )
      COMMON/GOMA/ INTIN(50),FPIN(50),INTOUT(50),FPOUT(100)
         EQUIVALENCE (FPOUT(1),DATA84(1))          ;OUTPUT TO HUD
            DIMENSION DATA84(43)
         EQUIVALENCE (FPOUT(66),UNPABYD3 )         ;UN VEC PIP BRST CS D3
         EQUIVALENCE (FPOUT(67),UNPABZD3 )         ;UN VEC PIP BRST CS D3
      COMMON/GVARB/ IVAR(50),FVAR(50),ICON(50),FCON(50)
         EQUIVALENCE (FVAR(4),PTALEX)              ;RANGE EST FROM FILTER
         EQUIVALENCE (FVAR(20),VTAGE (1))          ;VEL TGT/ATT GUN C.S. EST
            DIMENSION VTAGE (3)
            EQUIVALENCE (VTAGE (1),VTAGEX )
            EQUIVALENCE (VTAGE (2),VTAGEY )
            EQUIVALENCE (VTAGE (3),VTAGEZ )
         EQUIVALENCE (FVAR(23),SFTIGE(1))          ;SPEC FORCE TGT GUN C.S. ES
            DIMENSION SFTIGE(3)
            EQUIVALENCE (SFTIGE(1),SFTIGEX)
            EQUIVALENCE (SFTIGE(2),SFTIGEY)
            EQUIVALENCE (SFTIGE(3),SFTIGEZ)
         EQUIVALENCE (FVAR(28),RTF       )         ;INVERSE TOF BY LCOS
         EQUIVALENCE (FVAR(29),BC        )         ;BALLISTIC CURV BY LCOS
         EQUIVALENCE (FCON(20),RRH     )           ;INVERSE HARMON.RANGE
         EQUIVALENCE (FCON(23),PGHA(1))            ;POS GUN/HUD AC C. S.
            DIMENSION PGHA(3)
            EQUIVALENCE (PGHA(1),PGHAX)
            EQUIVALENCE (PGHA(2),PGHAY)
            EQUIVALENCE (PGHA(3),PGHAZ)
         EQUIVALENCE (FCON(26),XBIASHUD )          ;X-BIAS FOR HUD (RAD)
         EQUIVALENCE (FCON(27),YBIASHUD )          ;Y-BIAS FOR HUD (RAD)
         EQUIVALENCE (FCON(28),XSCALEHUD ) ;X-SCALE FAC FOR HUD (IN/RAD)
         EQUIVALENCE (FCON(29),YSCALEHUD ) ;Y-SCALE FAC FOR HUD (IN/RAD)
C
C-----INPUTS----------
C     VTAGEY,VTAGEZ     TARGET RELATIVE VELOCITY IN GUN COORDINATES
C                             (FT/SEC)
C     SFTIGEY,SFTIGEZ      TARGET SPECIFIC ACCELERATION IN GUN COORDINATE
C                             (FT/SEC/SEC)
C     PTALEX             RANGE TO TARGET (FT)
C     RTF               RECIPROCAL OF BULLET TIME OF FLIGHT (1/SEC)
C     BC                BALLISTIC CURVATURE
C
C-----OUTPUTS----------
C     UNPABYD3,UNPABZD3       Y AND Z COMPONENTS OF PIPPER POSITION
C                          RELATIVE TO BORESIGHT
C
C-----CONSTANTS----------
C     PGHAY,PGHAZ      POSITION OF GUN RELATIVE TO HUD IN A/C COORDINATES
C     RRH              RECIPROCAL OF HARMONIZATION RANGE
C
      IF(MODE.GE.10) GO TO 10
      MODE=10
      RETURN
10    CONTINUE
```

```
TF=1./RTF
RRNGE=1./PTALEX
UNPABYD3=-(VTAGEY+0.5*SFTIGEY*TF)*TF*RRNGE+(RRNGE-RRH)*PGHAY
UNPABZD3=-(VTAGEZ+0.5*SFTIGEZ*TF+BC)*TF*RRNGE+(RRNGE-RRH)*PGHAZ
DATA84(41)=(UNPABYD3+XBIASHUD)*XSCALEHUD
DATA84(42)=(-UNPABZD3+YBIASHUD)*YSCALEHUD
MODE=20
RETURN
END
```

```
      SUBROUTINE ACE
C
C     AFAL ACE GENERATED FOR EXPECTED NUMBER OF HITS
C     SEE AFAL TR-73-20
C
      COMMON/GMODE/MODEWORD(10),MODEOPT(10),MODEERR(10),SWBIT(2:16)
            LOGICAL SWBIT
         EQUIVALENCE (MODEWORD(6),MODE    )
      COMMON/GDMA/ INTIN(50),FPIN(50),INTOUT(50),FPOUT(100)
        EQUIVALENCE (FPIN(38),UNLAA(1)   )          :UNIT VEC. ALONG LOS
          DIMENSION UNLAA(3)
          EQUIVALENCE (UNLAA(1),UNLAAX)
          EQUIVALENCE (UNLAA(2),UNLAAY)
          EQUIVALENCE (UNLAA(3),UNLAAZ)
        EQUIVALENCE (FPOUT(88),UNPABYTR )         :UN VEC PIP BS CS TRGR
        EQUIVALENCE (FPOUT(89),UNPABZTR )         :UN VEC PIP BS CS TRGR
        EQUIVALENCE (FPOUT(90),ACEACTHITS)        :ACE ACTUAL HITS
        EQUIVALENCE (FPOUT(91),ACEPOSSHITS)       :ACE POSSIBLE HITS
      COMMON/GVARB/ IVAR(50),FVAR(50),ICON(50),FCON(50)
        EQUIVALENCE (FVAR(3),DTLOW      )         :DT FOR LOW RATE
        EQUIVALENCE (FVAR(4),PTALEX)            :RANGE EST FROM FILTER
        EQUIVALENCE (FCON(44),THBA    )   :B.S. ELEV. ANGLE
        EQUIVALENCE (FCON(45),SIBA    )   :B.S. AZIM. ANGGLE
        EQUIVALENCE (FCON(46),SIGT    )   :S.D.OF TARGET (FEET)
        EQUIVALENCE (FCON(47),SIGBS   )   :S.D. OF BUL. STR. (RAD)
        EQUIVALENCE (FCON(48),FIRERATE)   :FIRE RATE
      COMMON/LACE/ SQ2PI,E,AZOLD,ELOLD,W2,BULN
         LOGICAL RBULATE
      DATA SQ2PI/2.5066283/
      DATA E/2.7182818/
C
C-----INPUTS--------------
C
C     AZ              THE AZIMUTH POSITION OF THE COMPUTED
C               BULLETS AT TARGET RANGE WITH RESPECT TO THE
C               TARGET      (RAD)
C     EL              THE ELEVATION POSITION OF THE COMPUTED
C               BULLETS AT TARGET RANGE WITH RESPECT TO THE
C               TARGET      (RAD)
C     PTALEX           RANGE OF THE TARGET  (FEET)
C     IFLAG           INITIALIZATION FLAG (IFLAG=1---RUN)
C                                         (IFLAG.NE.1---INITIALIZE)
C
C
C
C-----OUTPUT--------------
C     EXPHTS           EXPECTED NUMBER OF HITS PER COMPUTER CYCLE
C
C
C
C-----INPUT CONSTANTS-----------
C     BULN              THE NUMBER OF BULLETS AT TARGET RANGE
C               PER COMPUTER CYCLE TIME  (FIRE RATE * CYCLE TIME)
C     SIGT              STANDARD DEVIATION OF THE TARGET  (FEET)
C     SIGBS             STANDARD DEVIATION OF THE BULLET
C               STREAM OR BULLET DISPERSION (RAD)
C
```

```
      C
      C
   60 C-----INTERNAL CONSTANTS-------
      C     SQ2PI             SQUARE ROOT OF (2.*3.1415926536)
      C     E                 EXPO. NUMBER  (2.7182818285)
      C     AZOLD             PREVIOUS VALUE OF AZIMUTH POSITION (RAD)
      C     ELOLD             PREVIOUS VALUE OF ELEVATION POSITION (RAD)
   65 C     W2                SIGBS*SIGBS  (RAD**2)
      C
      C
      C
            AZ=UNPABYTR-(-SIBA*UNLAAX+UNLAAY)
   70       EL=UNPABZTR-(THBA*UNLAAX+UNLAAZ)
            IF(MODE.GE.10) GO TO 50
      C
      C     DEFINE INTERNAL CONSTANTS
      C
   75       W2=SIGBS*SIGBS
            AZOLD=AZ
            ELOLD=EL
            RULN=FIRERATE*DTLOW
            RULATR=.FALSE.
   80       ACEPOSSHITS=0
            ACEACTHITS=0.
            MODE=MODE+10
      C
      C     COMPLETE INITIALIZATION
   85 C
            RETURN
      C
      C     BEGIN MAIN LOOP TO COMPUTE EXPHTS
      C
   90 C     COMPUTE RELATIVE MOTION VECTOR
      C
      50    AZL=AZOLD-AZ
            ELL=ELOLD-EL
            XMAGL2=AZL*AZL+ELL*ELL
   95       XMAGL=SQRT(XMAGL2)
      C
      C     COMPUTE ANGLE
      C
            XNUM=AZL*AZOLD+ELL*ELOLD
  100       RMAG2=AZOLD*AZOLD+ELOLD*ELOLD
            RMAG=SQRT(RMAG2)
            IF(XMAGL.EQ.0.) XMAGL=.000001
            IF(RMAG.EQ.0.) RMAG=.000001
            XDEM=XMAGL*RMAG
  105       COSDL=XNUM/XDEM
            IF(COSDL.GT.1.) COSDL=1.
            IF(COSDL.LT.-1.) COSDL=-1.
      C
      C     COMPUTE MISS VECTOR OF TARGET WITH RESPECT TO RELATIVE MOTION
  110 C     VECTOR
      C
            SINDL2=1.-COSDL*COSDL
            SINDL=SQRT(SINDL2)
            XT=RMAG*SINDL
```

```
      115               YT=RMAG*COSDL-XMAGL/2.
                        ST=SIGT/PTALTX
                        S2=ST*ST
                C
                C       BEGIN COMPUTATION OF EXPECTED NUMBER OF HITS
      120       C
                        SW=S2+W2
                        EE=-(XT*XT)/(2.*SW)
                        XNUM=BULN*S2*SQ2PI*E**EE
                        XDEM=XMAGL*SQRT(SW)
      125               ERPL=(XMAGL/2.-YT)/ST
                        ERMI=(-XMAGL/2.-YT)/ST
                C
                C       COMPUTE EXPECTED NUMBER OF HITS
                C
      130               EXPHTS=XNUM*(ERF(ERPL)-ERF(ERMI))/XDEM
                        ACEPOSSHITS=ACEPOSSHITS+EXPHTS
                        IF(BULATR) ACEACTHITS=ACEACTHITS+EXPHTS
                C
                C       SAVE AZ AND EL FOR NEXT COMPUTATION
      135       C
                        AZOLD=AZ
                        ELOLD=EL
                        IF(MODE.LT.20) MODE=MODE+10
                        RETURN
      140               END
```

160

```
      FUNCTION ERF(X)
C
C     ERROR FUNCTION CURVE FIT TO A THIRD DEGREE EQUATION
C     OVER THE INTERVAL -3.5 TO 3.5
C     OUTSIDE THIS INTERVAL  ERF ASSIGNED EITHER -.5 OR .5
C
      LOGICAL NEG
      NEG=.FALSE.
      IF(X.LT.0.) NEG=.TRUE.
      X=ABS(X)
      IF(X.GE.3.5) GO TO 100
      IF(X.LE..2) GO TO 50
      ERF=-.01322338+X*(.49613045+X*(-.15942666+.01698544*X))
      GO TO 150
50    ERF=.39529855*X
      GO TO 150
100   ERF=.5
150   IF(NEG) ERF=-ERF
      RETURN
      END
```